Synthesis of dependable and self-driving communication networks

Stefan Schmid
@FRIDA 2025

Acknowledgements:

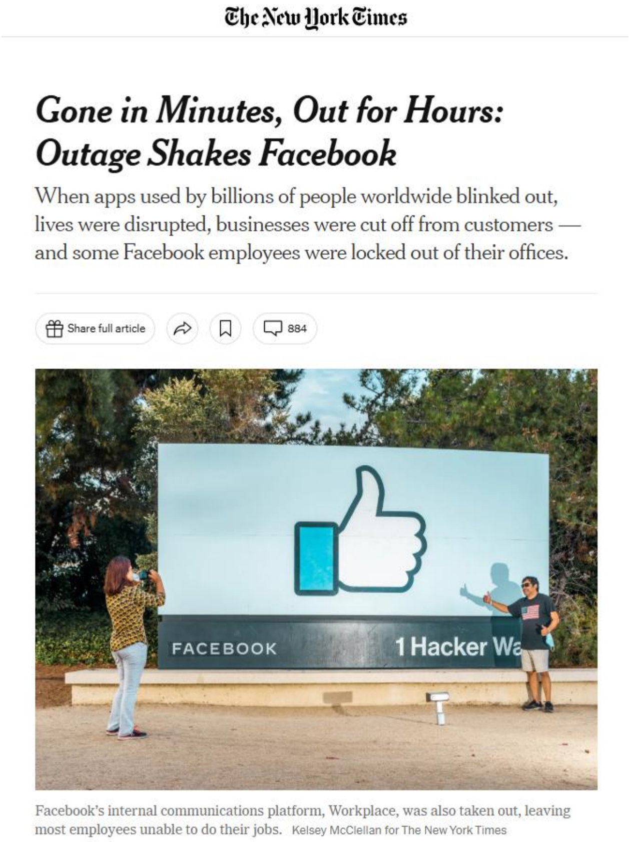
W|W|T|F

erc
European Research Council
Established by the European Commission

Networks:

Critical Infrastructure

- If networks break, it can have knock-on effects
- For example, Facebook outage in 2021: not only took down their social networking site, but also Instagram, WhatsApp, …
- … and their own internal systems, which manage the doors: engineers had to break into their own buildings to bring the network back up



The New York Times

Gone in Minutes, Out for Hours: Outage Shakes Facebook

When apps used by billions of people worldwide blinked out, lives were disrupted, businesses were cut off from customers — and some Facebook employees were locked out of their offices.

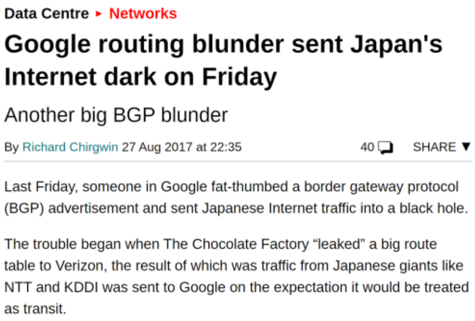
Facebook's internal communications platform, Workplace, was also taken out, leaving most employees unable to do their jobs. Kelsey McClellan for The New York Times

Credits: Nate Foster

The Challenge: Most Outages due to Human Errors

Human Errors

Countries disconnected



Data Centre ▸ Networks

Google routing blunder sent Japan's Internet dark on Friday

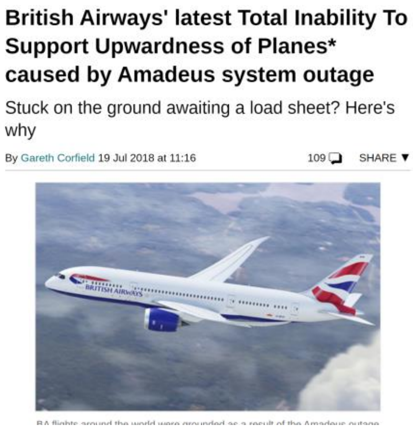
Another big BGP blunder

By Richard Chirgwin 27 Aug 2017 at 22:35 40 SHARE ▼

Last Friday, someone in Google fat-thumbed a border gateway protocol (BGP) advertisement and sent Japanese Internet traffic into a black hole.

The trouble began when The Chocolate Factory "leaked" a big route table to Verizon, the result of which was traffic from Japanese giants like NTT and KDDI was sent to Google on the expectation it would be treated as transit.

Passengers stranded



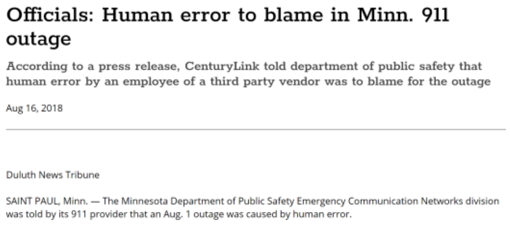
British Airways' latest Total Inability To Support Upwardness of Planes* caused by Amadeus system outage

Stuck on the ground awaiting a load sheet? Here's why

By Gareth Corfield 19 Jul 2018 at 11:16 109 SHARE ▼

BA flights around the world were grounded as a result of the Amadeus outage

Even 911 affected



Officials: Human error to blame in Minn. 911 outage

According to a press release, CenturyLink told department of public safety that human error by an employee of a third party vendor was to blame for the outage

Aug 16, 2018

Duluth News Tribune

SAINT PAUL, Minn. — The Minnesota Department of Public Safety Emergency Communication Networks division was told by its 911 provider that an Aug. 1 outage was caused by human error.

Even tech-savvy companies struggle:

Go Daddy.com github SOCIAL CODING United Airlines amazon web services

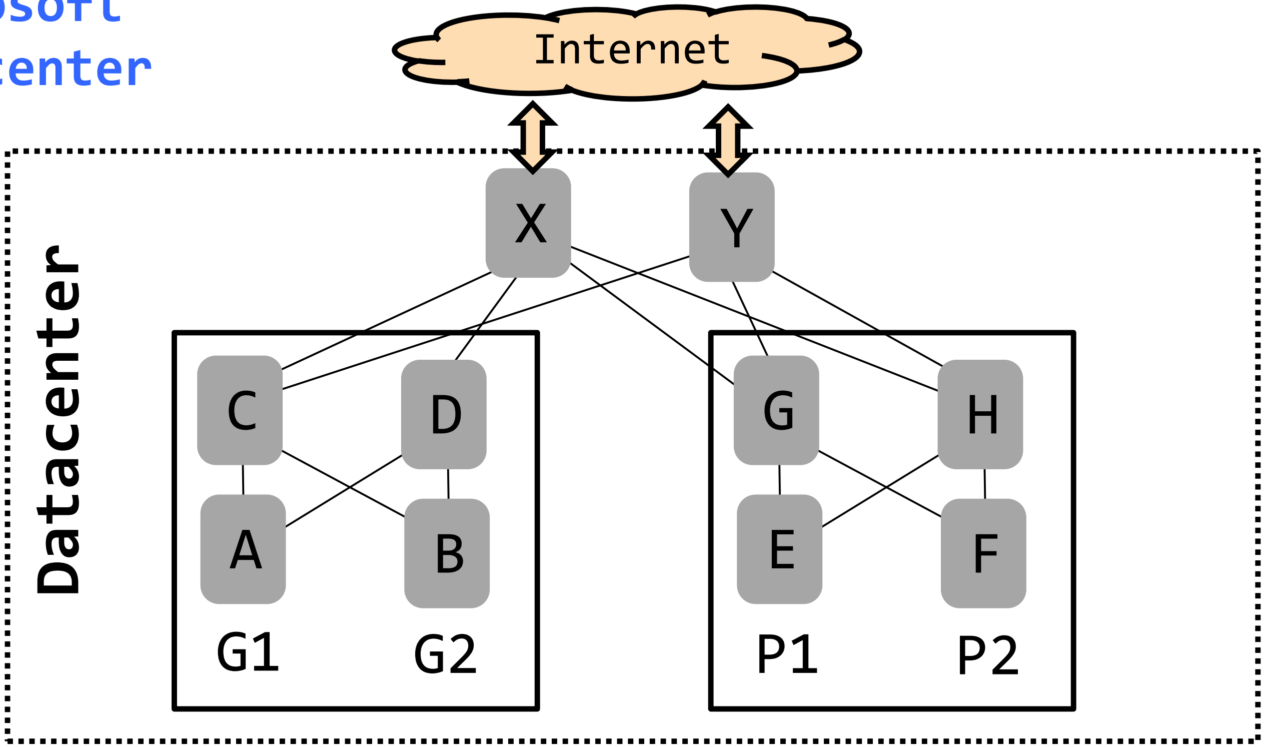
Mainly: human errors!

Slide credits: Nate Foster and Laurent Vanbever

A Reason: Complexity

Especially Under Failures (Policy Compliance)

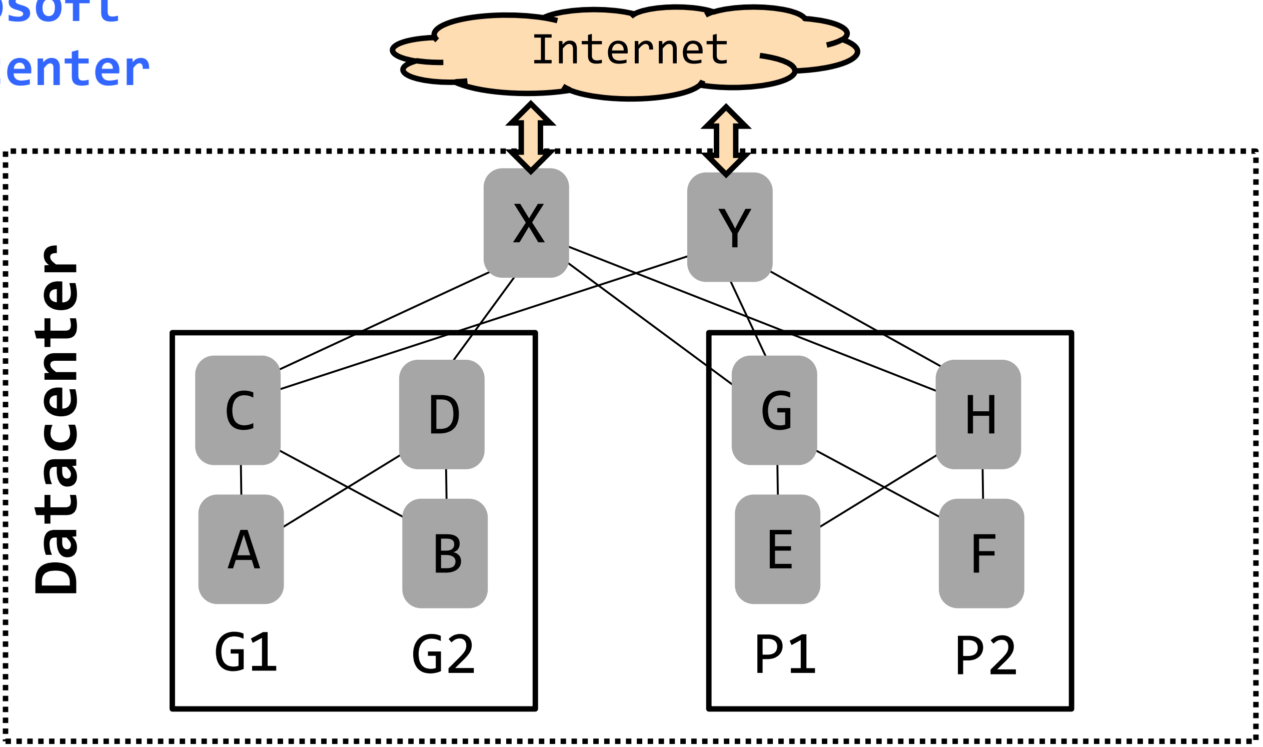
Example: BGP in **Microsoft datacenter**



A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**



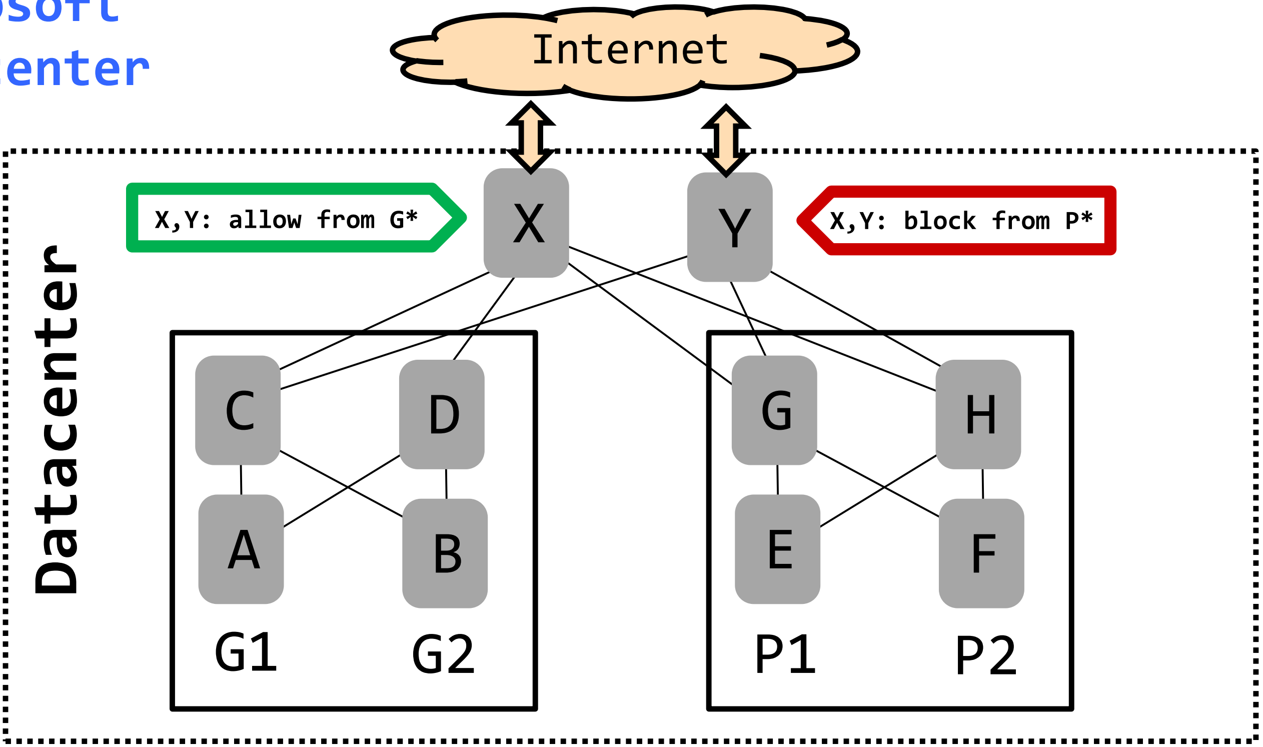
Cluster with globally reachable services

Cluster with internally accessible services

A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in Microsoft datacenter



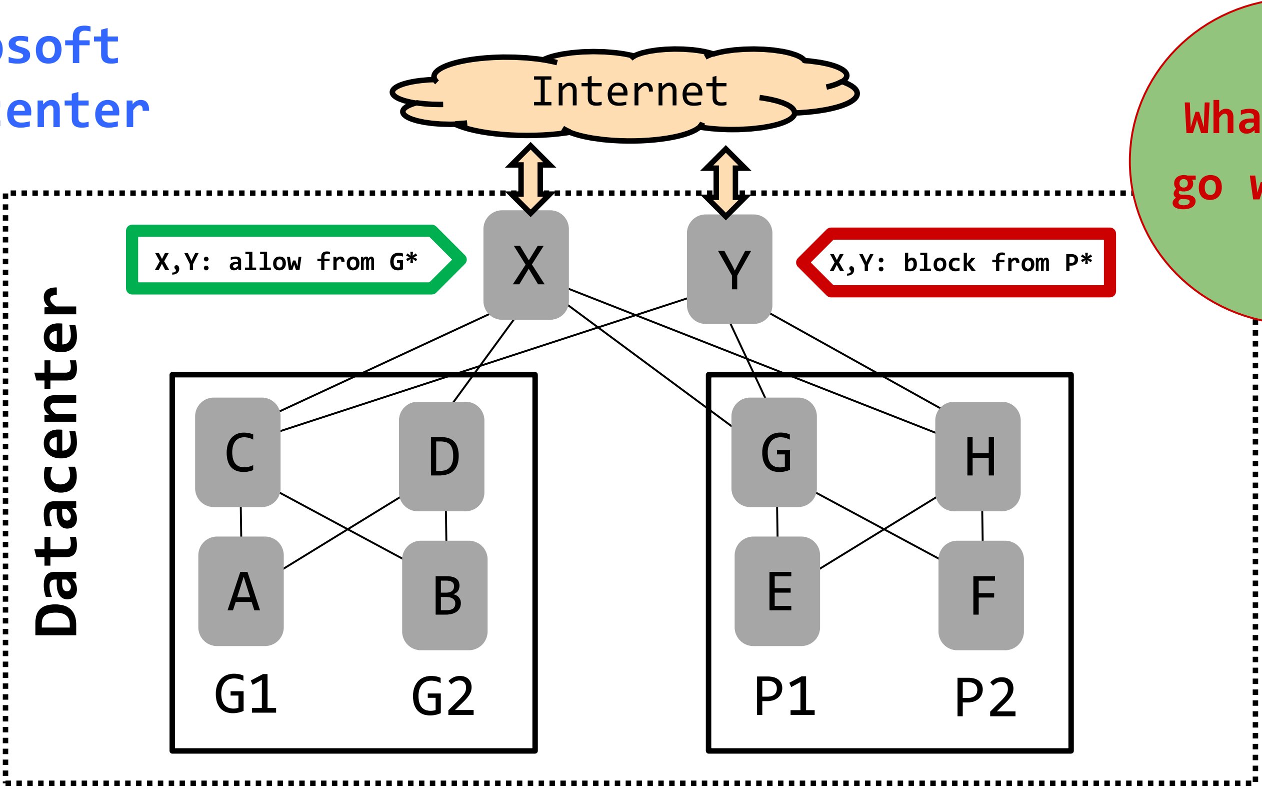
Cluster with globally reachable services

Cluster with internally accessible services

A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in Microsoft datacenter



What can go wrong?

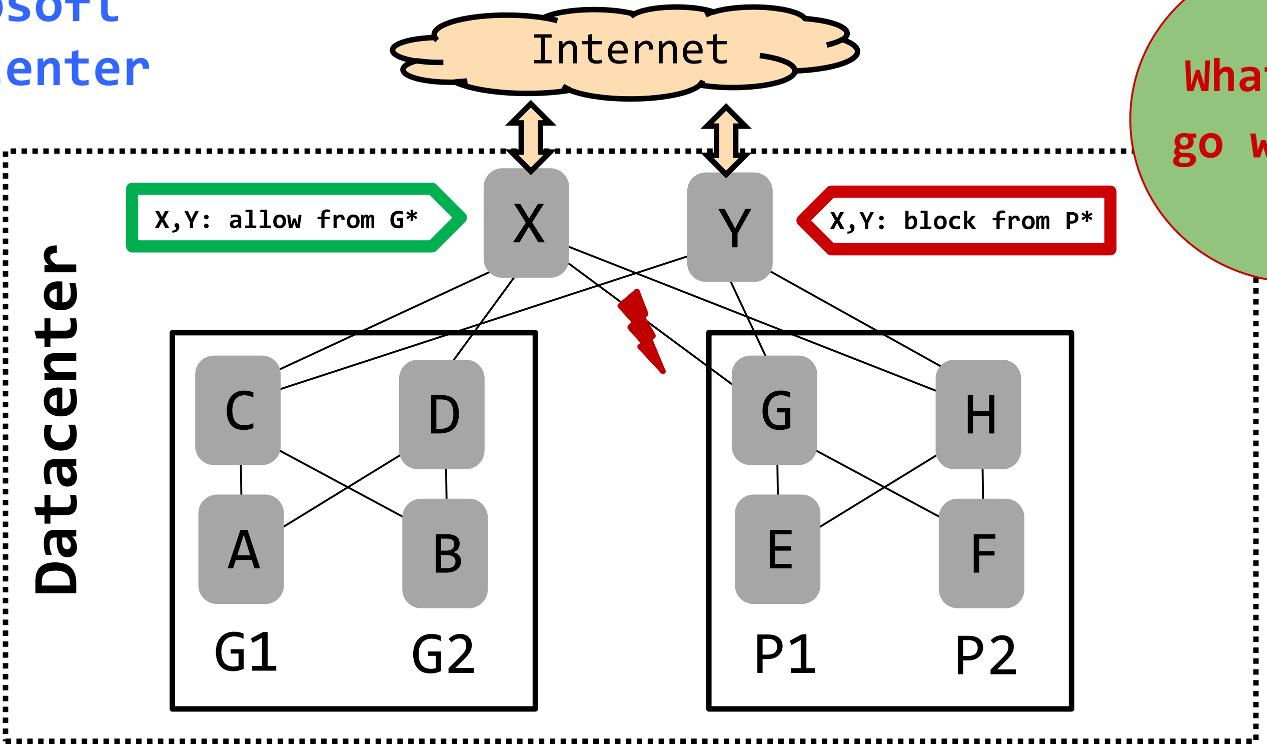
Cluster with globally reachable services

Cluster with internally accessible services

A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in **Microsoft datacenter**

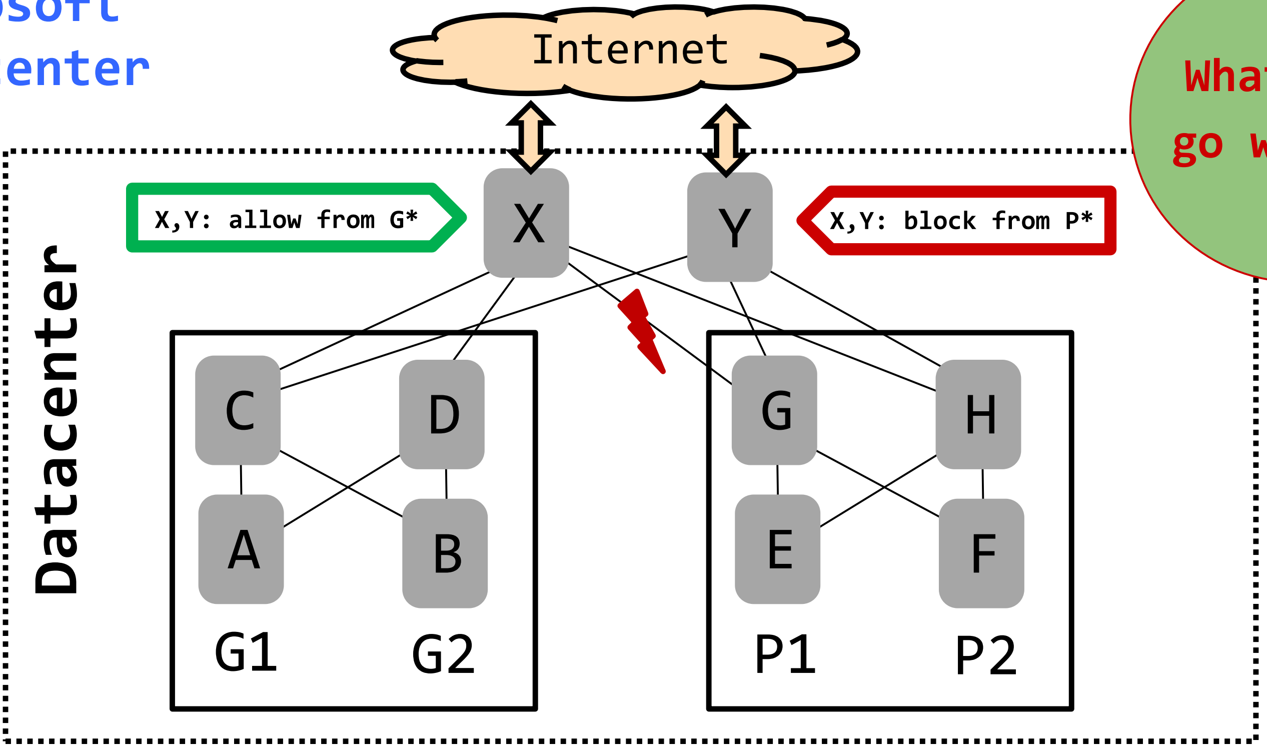


What can go wrong?

A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in Microsoft datacenter



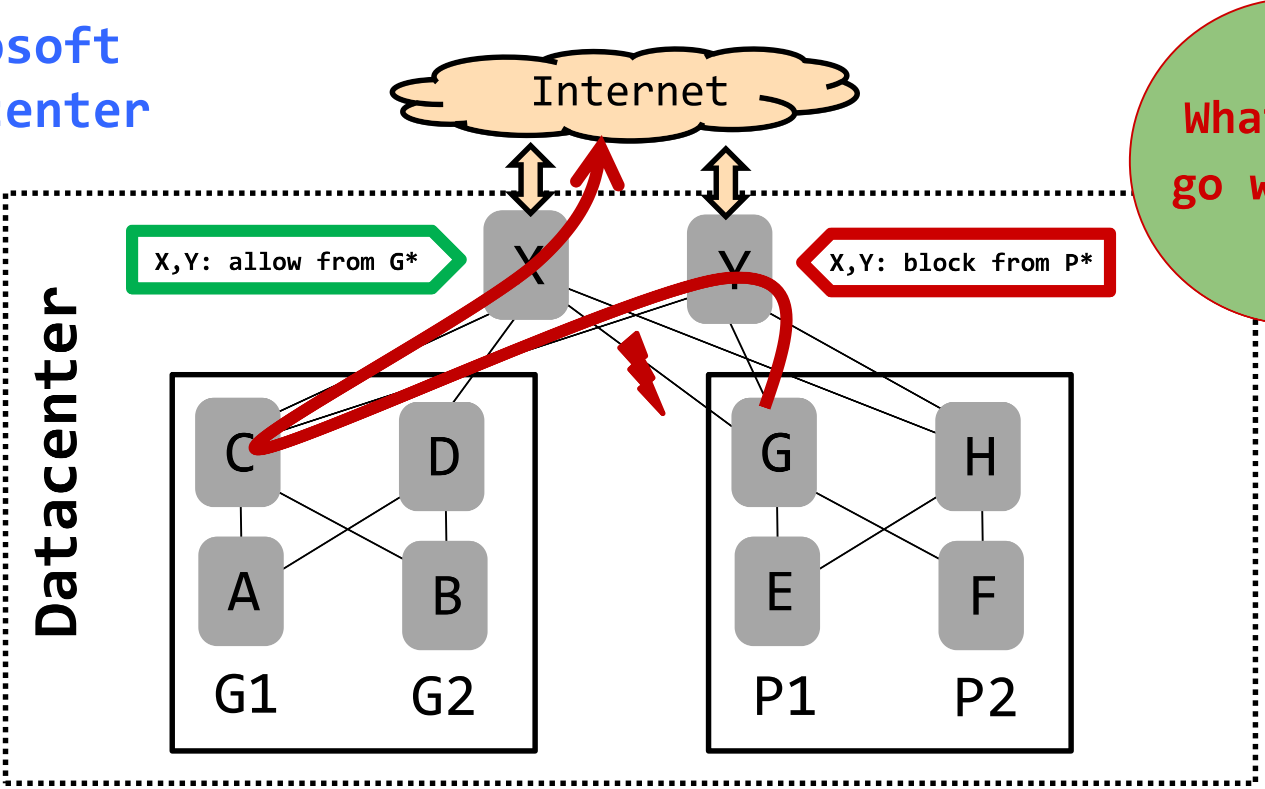
What can go wrong?

If link (G,X) fails and traffic from G is rerouted via Y and C to X:
X announces (does not block) G and H as it comes from C. (Note: BGP.)

A Reason: Complexity

Especially Under Failures (Policy Compliance)

Example: BGP in Microsoft datacenter



What can go wrong?

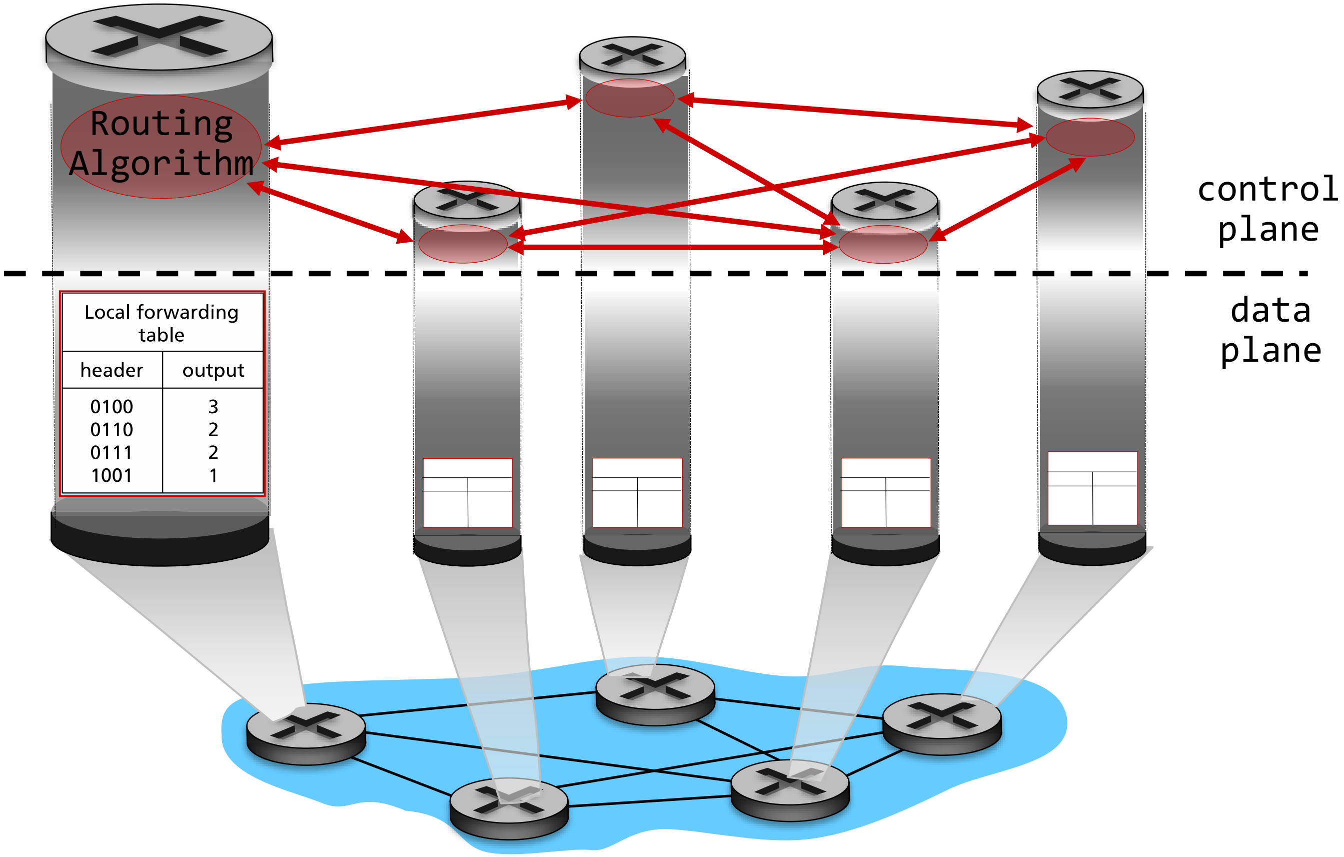
If link (G,X) fails and traffic from G is rerouted via Y and C to X:
X announces (does not block) G and H as it comes from C. (Note: BGP.)

Particularly Difficult

Fast Rerouting

Particularly Difficult

Local Fast Rerouting

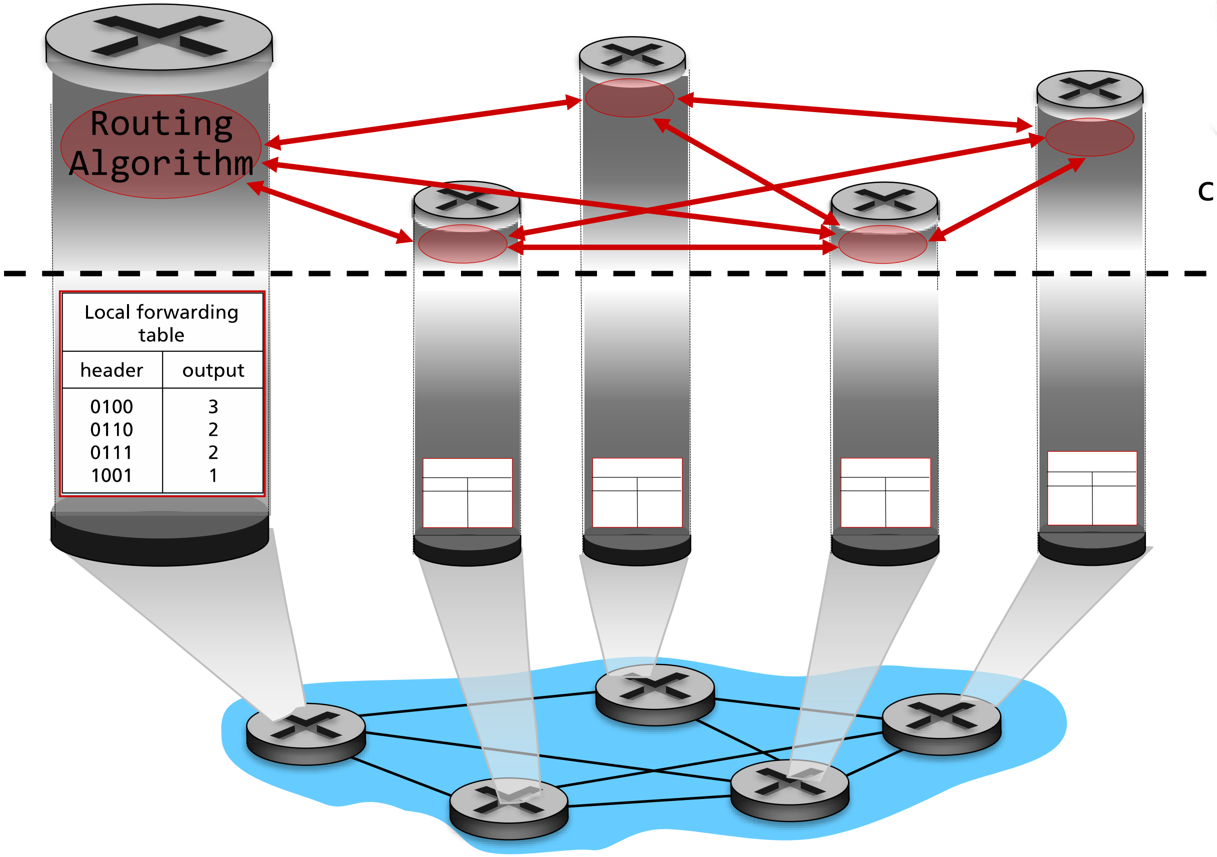


Local forwarding table

header	output
0100	3
0110	2
0111	2
1001	1

Particularly Difficult

Local Fast Rerouting



Local forwarding table	
header	output
0100	3
0110	2
0111	2
1001	1

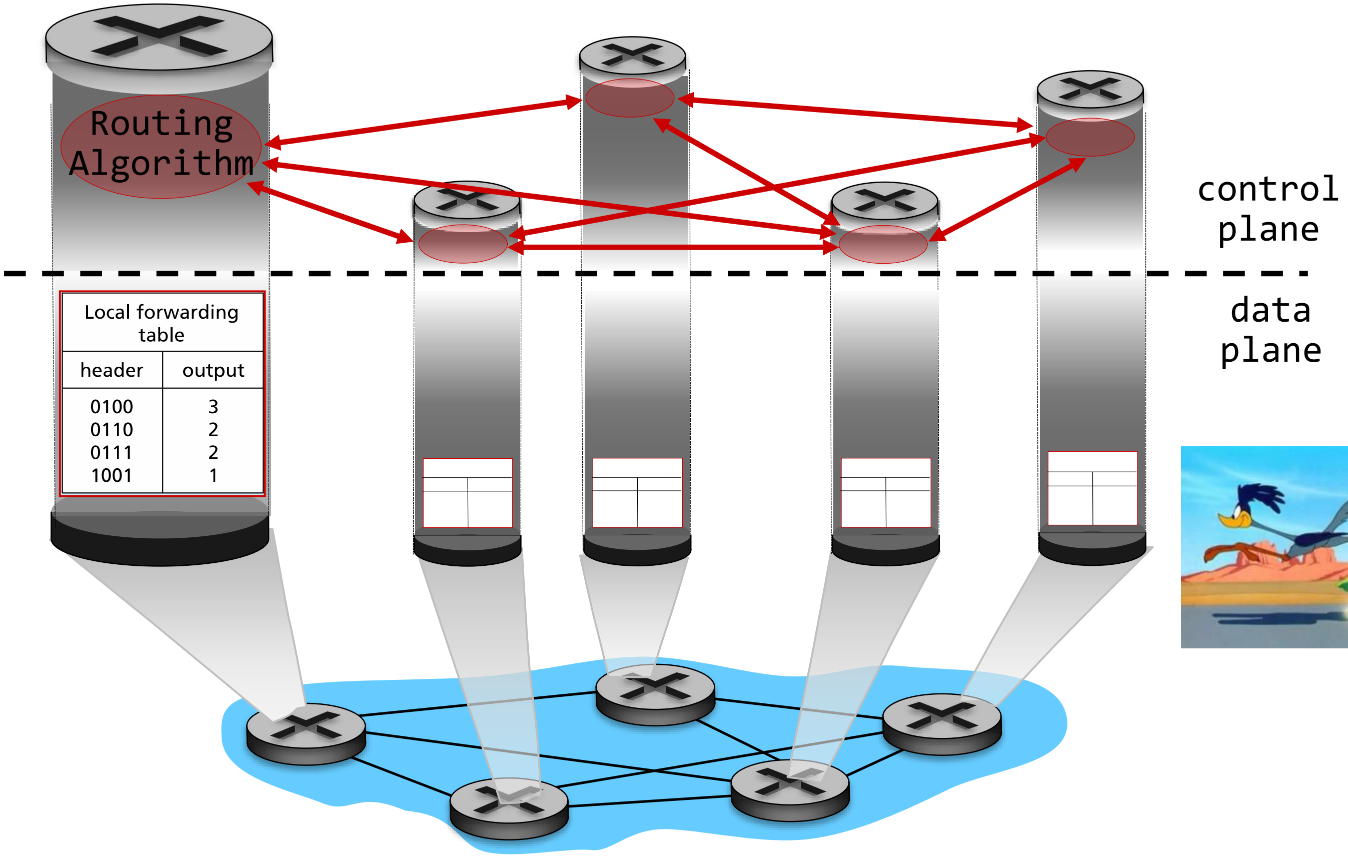
control plane

data plane

Slow but global

Particularly Difficult

Local Fast Rerouting



Local forwarding table

header	output
0100	3
0110	2
0111	2
1001	1

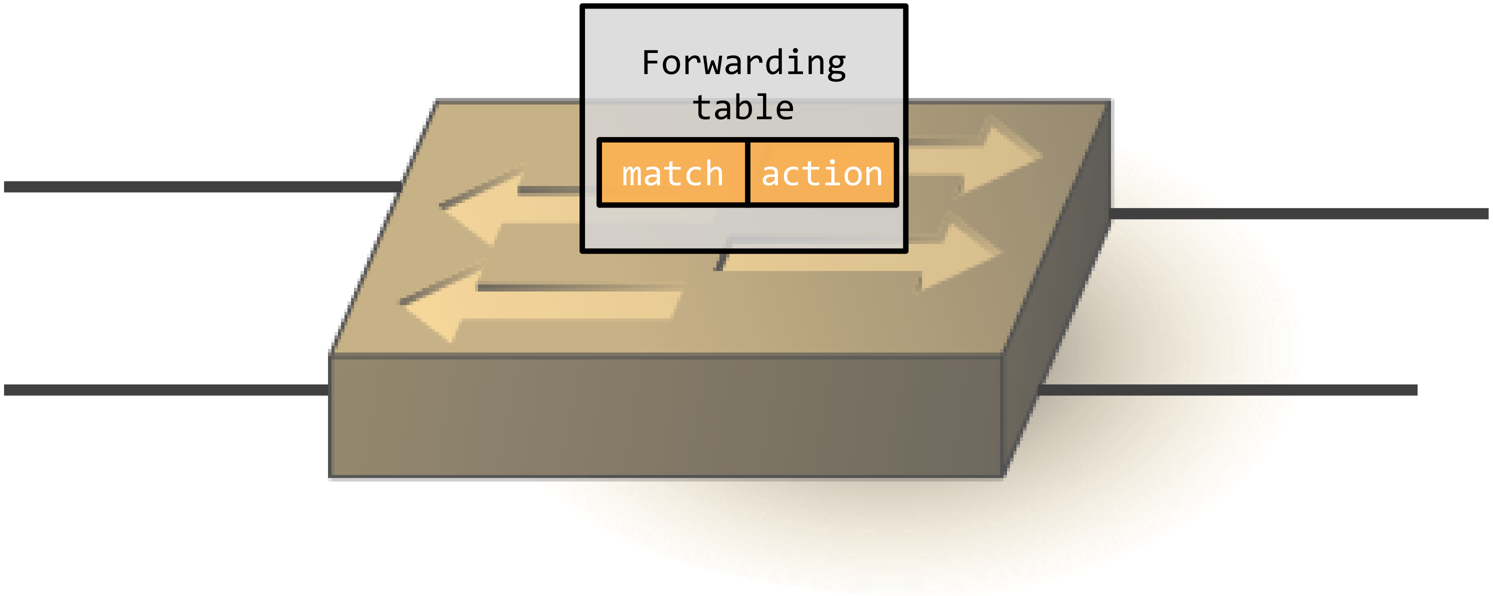
Fast but local

Information at Switch for

Local Decision Making?

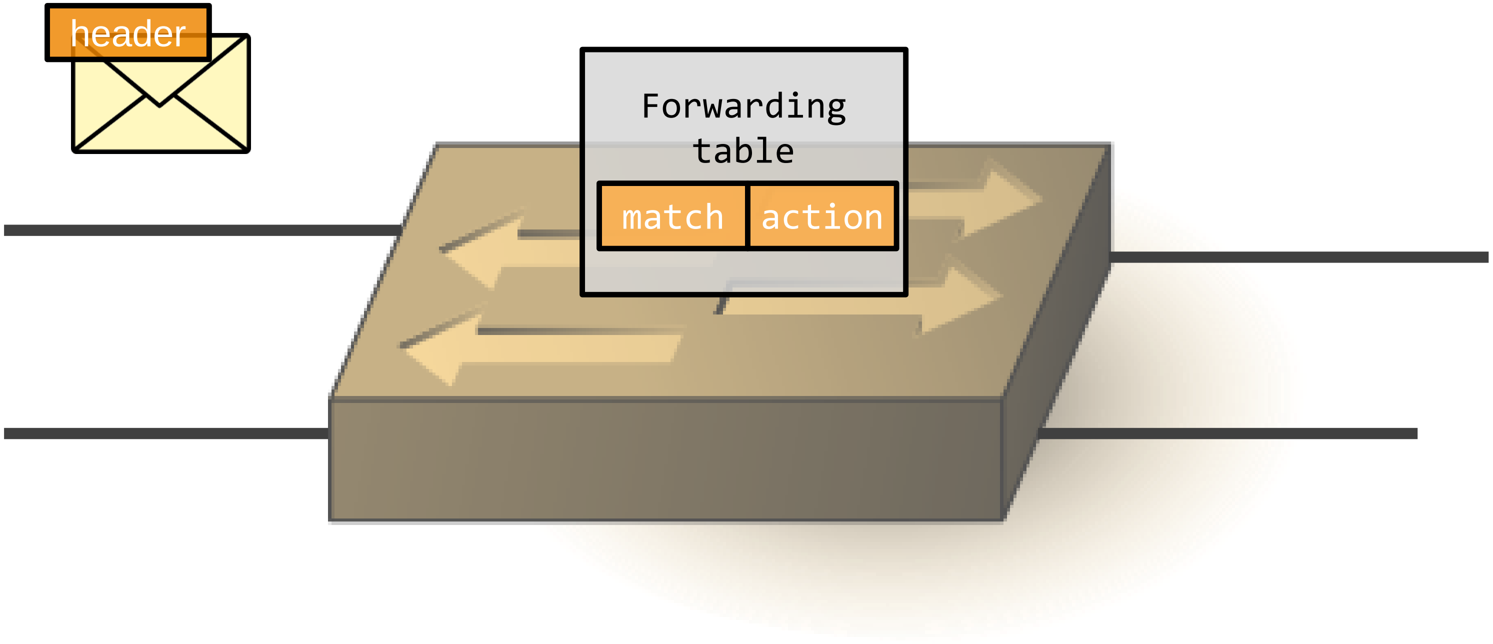
Information at Switch for
Local Decision Making?

⇢ Nodes locally store a forwarding Match -> Action table



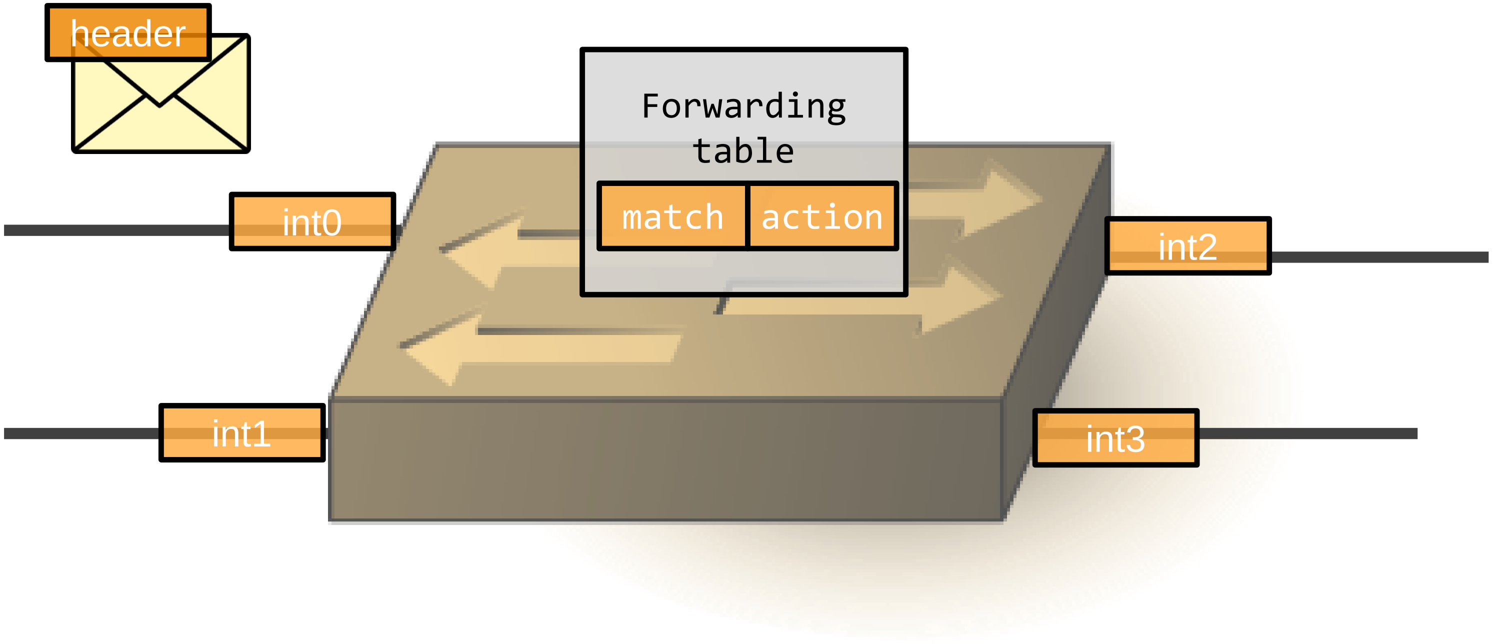
Information at Switch for
Local Decision Making?

⇢ The Packet Header (e.g., source, destination)



Information at Switch for Local Decision Making?

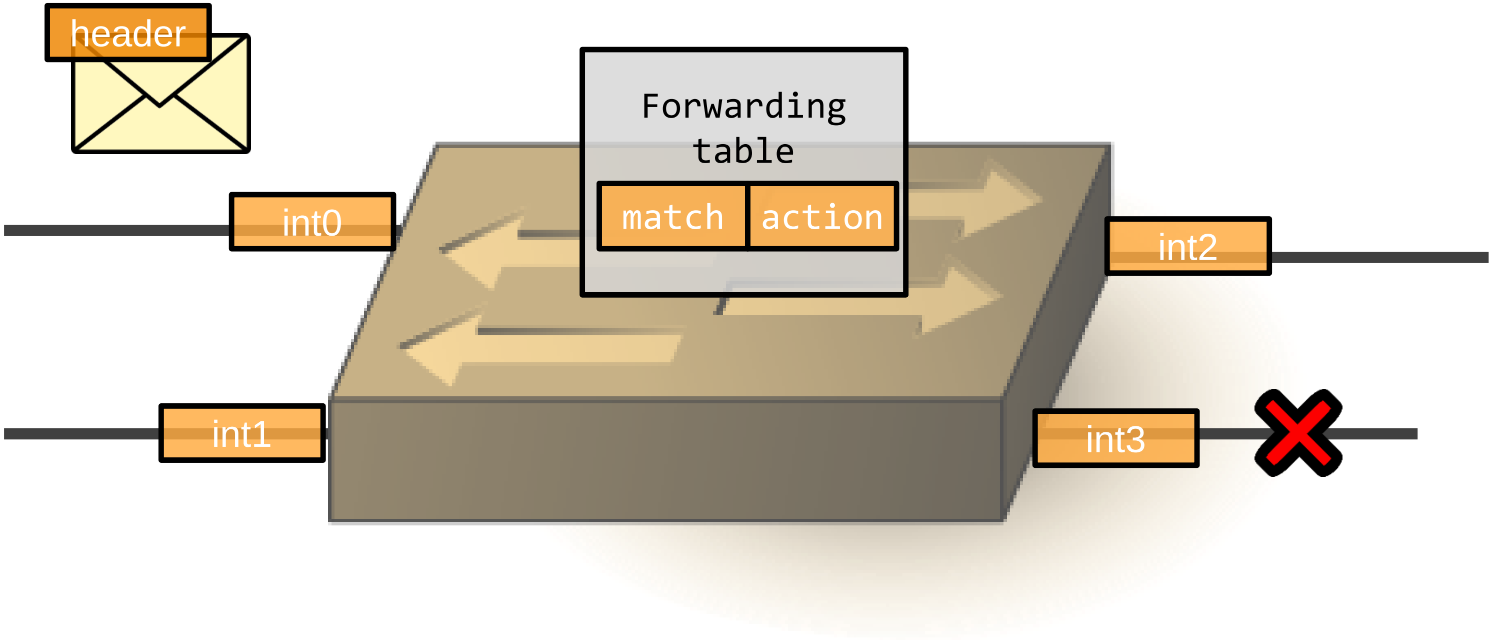
⇢ The Inport of the received packet



Information at Switch for

Local Decision Making?

⇢ Which incident links failed



Objective

What-if Analysis & Synthesis

- … for robust networks tolerating many link failures.
- Verification: Are the current forwarding rules policy compliant (reachability, waypoint traversal) even under failures?
- Synthesis: Can we pre-install local fast failover rules which ensure reachability under multiple failures?
- In general: How many failures can be tolerated by static forwarding tables?

Two fundamental

Notions of Resilience

Ideal resilience

Given a *k*-connected graphs, fast reroute can tolerate *any k-1 link failures.*

Perfect resilience

Fast reroute can tolerate any failures as long as the unterlying network is *physically connected*.

⇢ What is the difference? Which is stronger?

A big open challenge

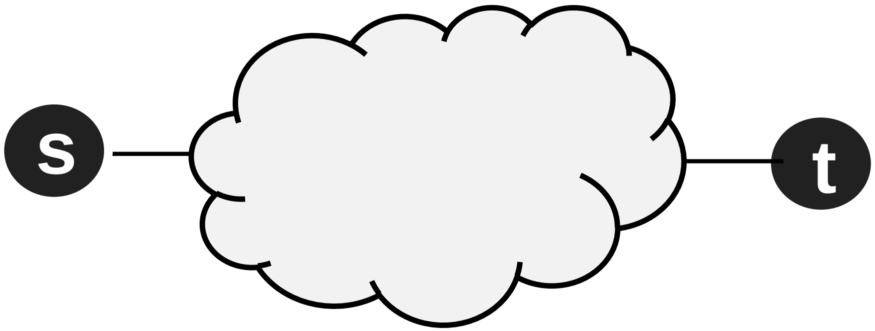
Ideal Resilience

- Given a k-connected network: how many link failures can a fast re-routing mechanism tolerate? Conjecture: k-1.
- Assume: cannot change header, but can match inport, src and dst

A big open challenge

Ideal Resilience

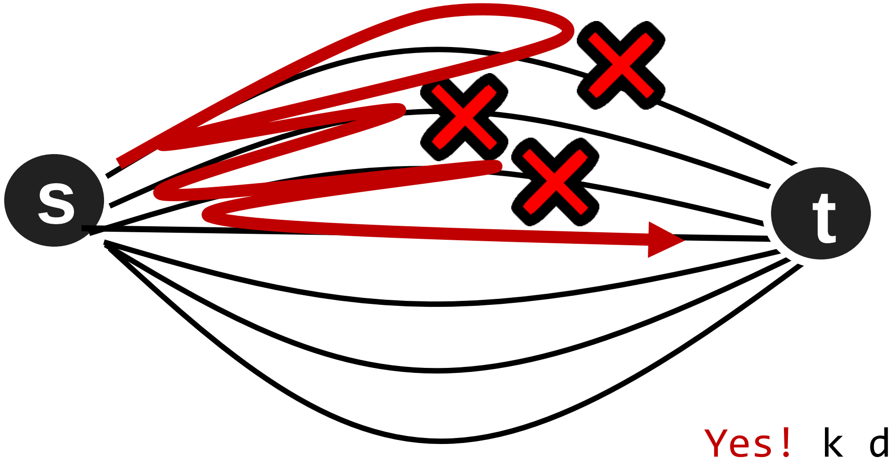
- Given a k-connected network: how many link failures can a fast re-routing mechanism tolerate? Conjecture: k-1.
- Assume: cannot change header, but can match inport, src and dst



A big open challenge

Ideal Resilience

- Given a k-connected network: how many link failures can a fast re-routing mechanism tolerate? Conjecture: k-1.
- Assume: cannot change header, but can match inport, src and dst



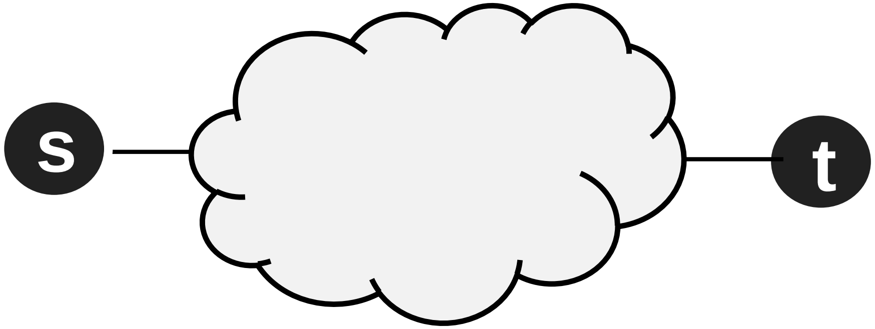
Yes! k disjoint paths: try one after the other, routing *back to source* each time.

A big open challenge

Ideal Resilience

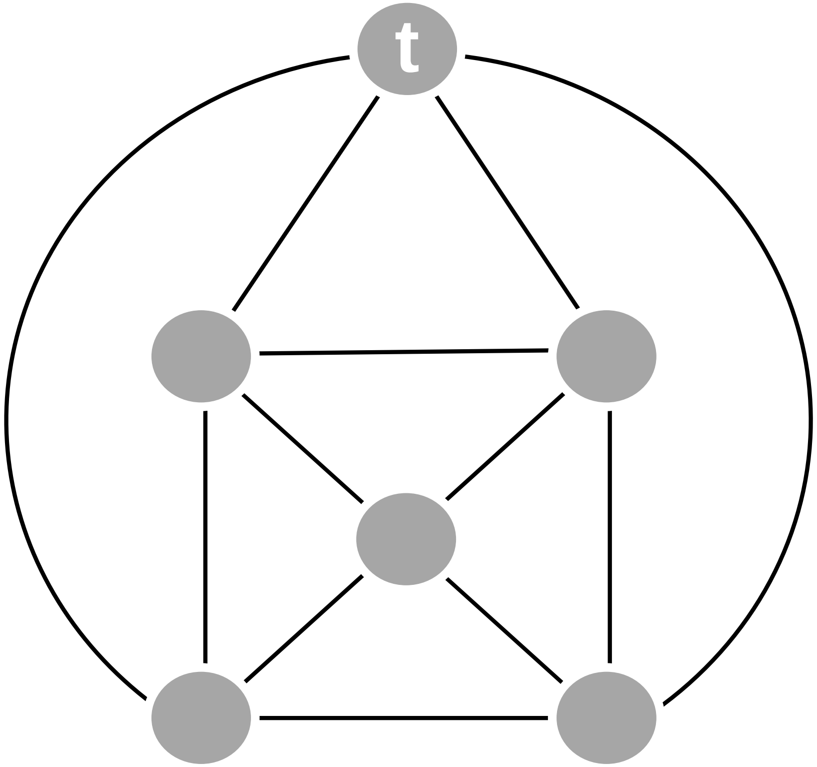
- Given a k-connected network: how many link failures can a fast re-routing mechanism tolerate? Conjecture: k-1.
- Assume: cannot change header, but can match inport, ✗ and dst

What if I cannot match source?! Open conjecture.



State-of-the-Art Approach for Ideal Resilience

Spanning Arborescences



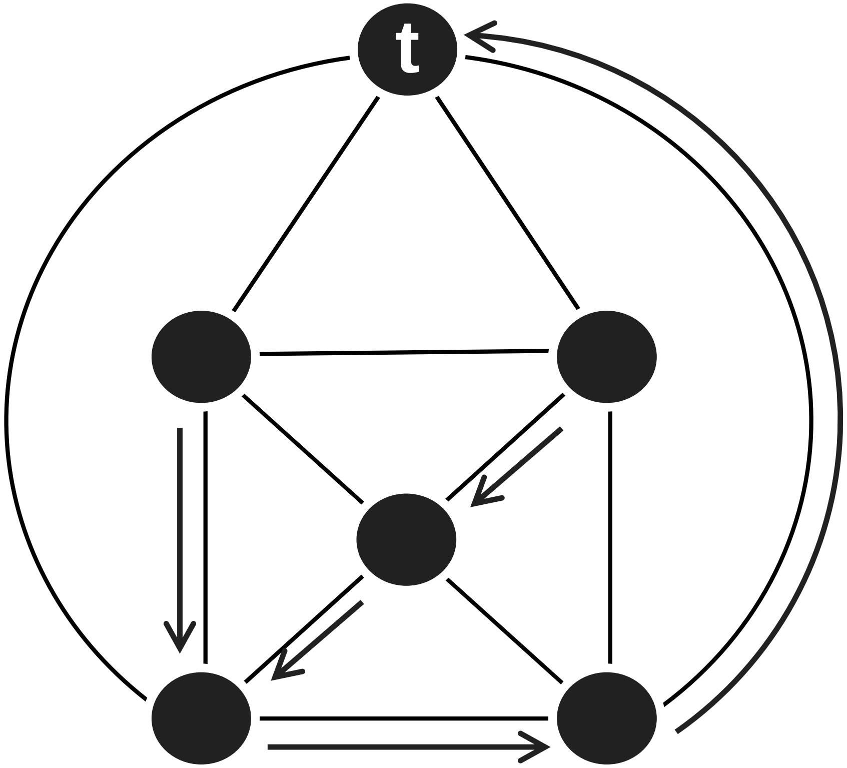
⇢ Fact: k-connected network has k-arborescence decomposition

⇢ Basically disjoint spanning trees directed to destination

State-of-the-Art Approach for Ideal Resilience

Spanning Arborescences

Arborescences



⇢ Fact: k-connected network has k-arborescence decomposition

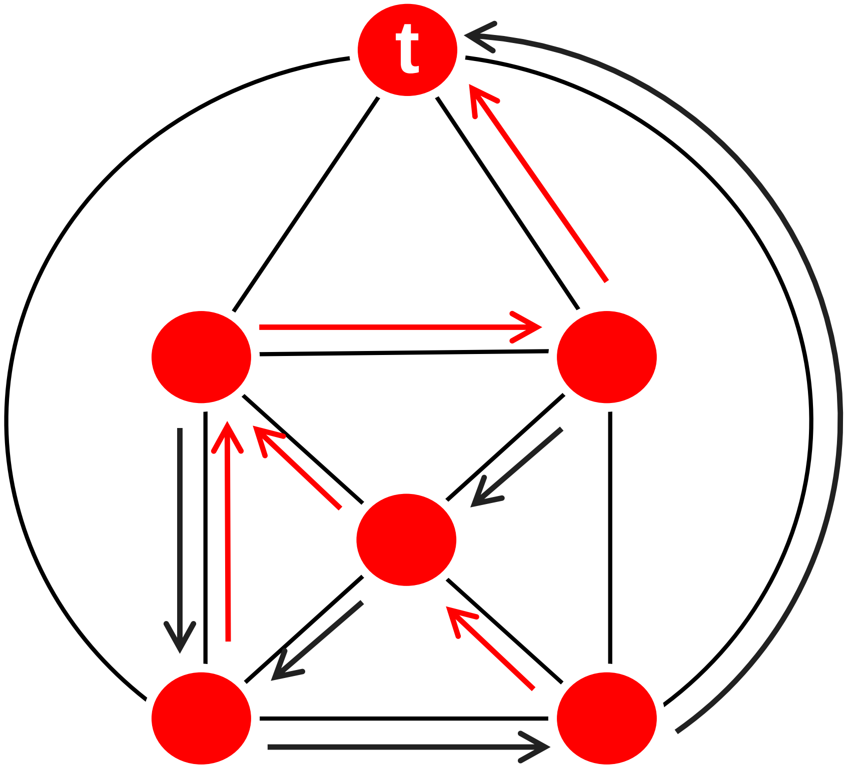
⇢ Basically disjoint spanning trees directed to destination

State-of-the-Art Approach for Ideal Resilience

Spanning Arborescences

Arborescences

1 2



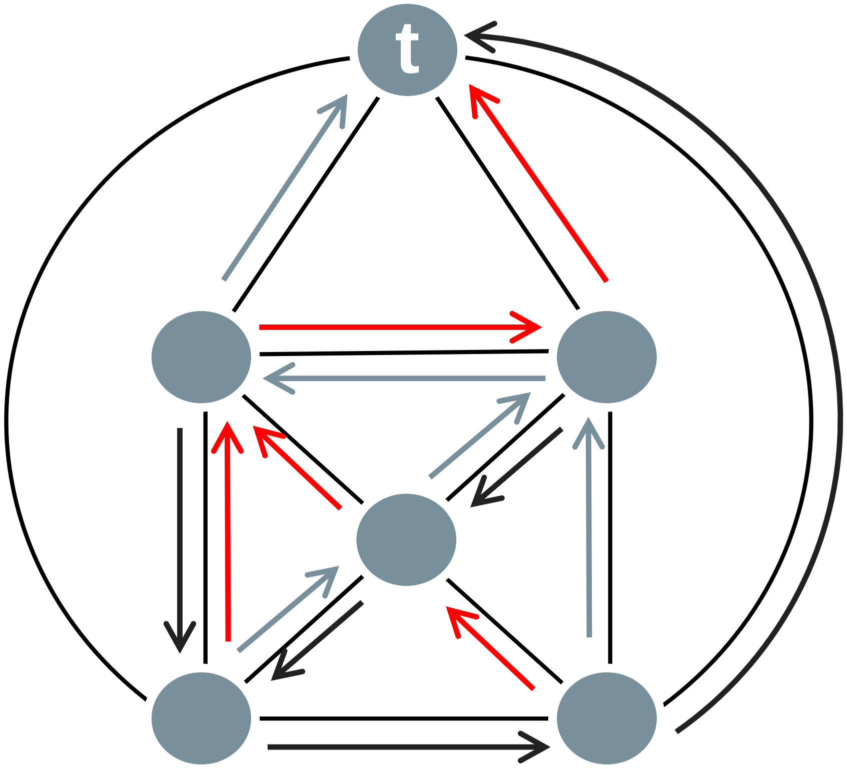
- Fact: k-connected network has k-arborescence decomposition
- Basically disjoint spanning trees directed to destination

State-of-the-Art Approach for Ideal Resilience

Spanning Arborescences

Arborescences

1 2 3



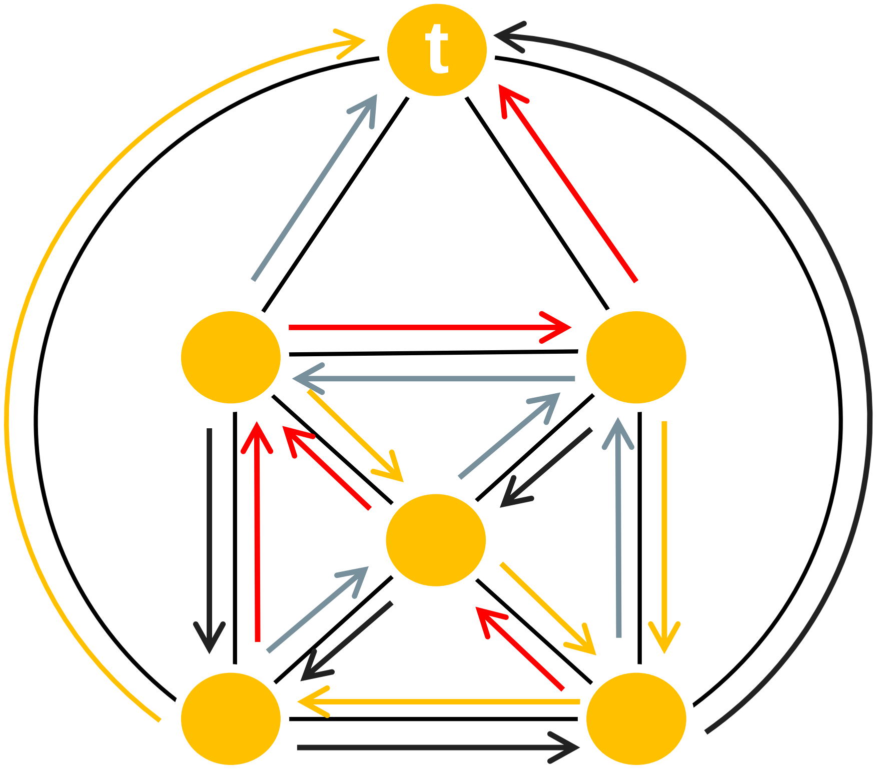
- Fact: k-connected network has k-arborescence decomposition
- Basically disjoint spanning trees directed to destination

State-of-the-Art Approach for Ideal Resilience

Spanning Arborescences

Arborescences

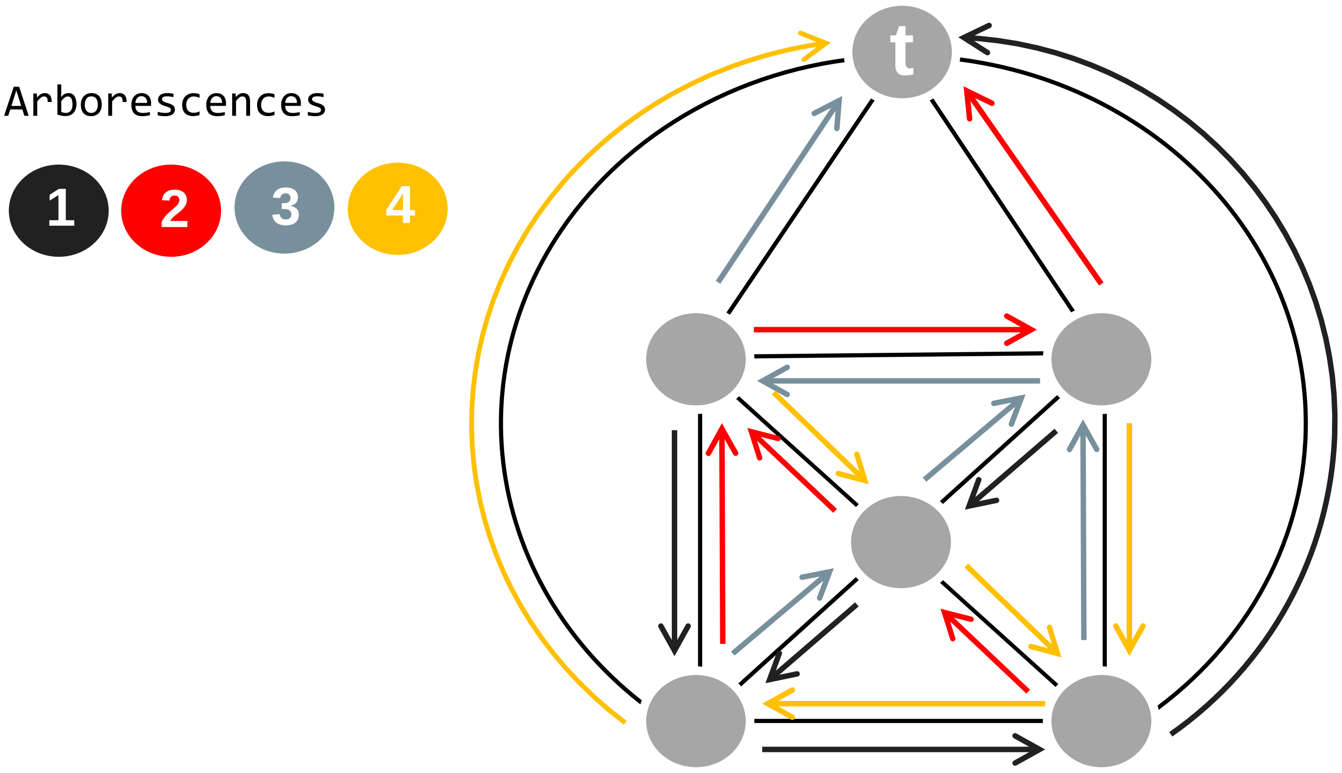
1 2 3 4



- ⇢ Fact: k-connected network has k-arborescence decomposition
- ⇢ Basically disjoint spanning trees directed to destination

State-of-the-Art Approach for Ideal Resilience

Spanning Arborescences



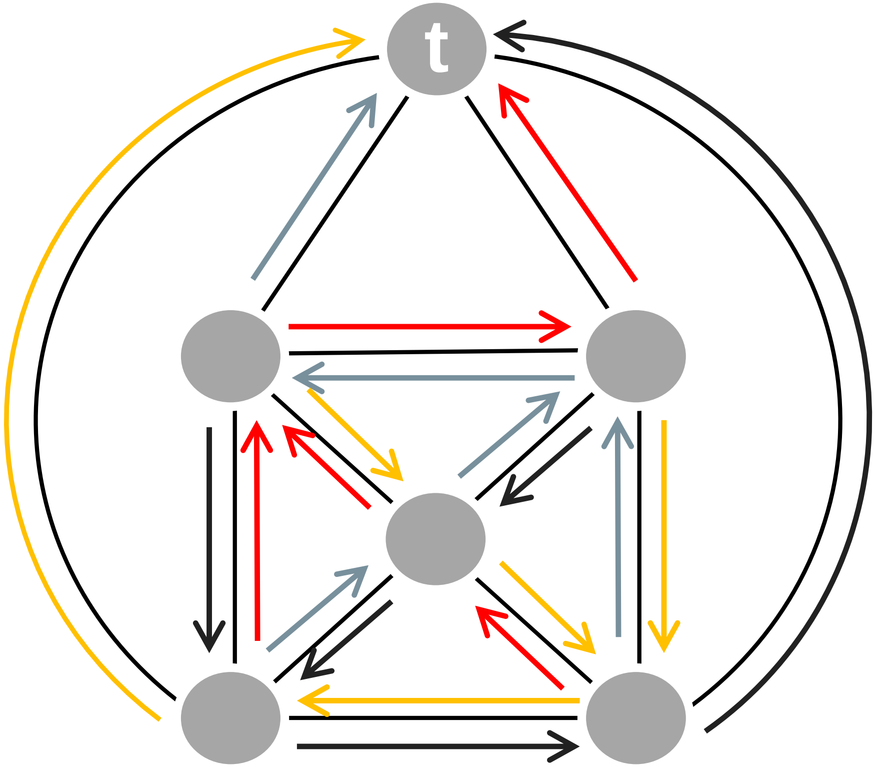
- Fact: k-connected network has k-arborescence decomposition
- Basically disjoint spanning trees directed to destination

(k/2-1)-resilient with circular

Arborescence Routing

Arborescences

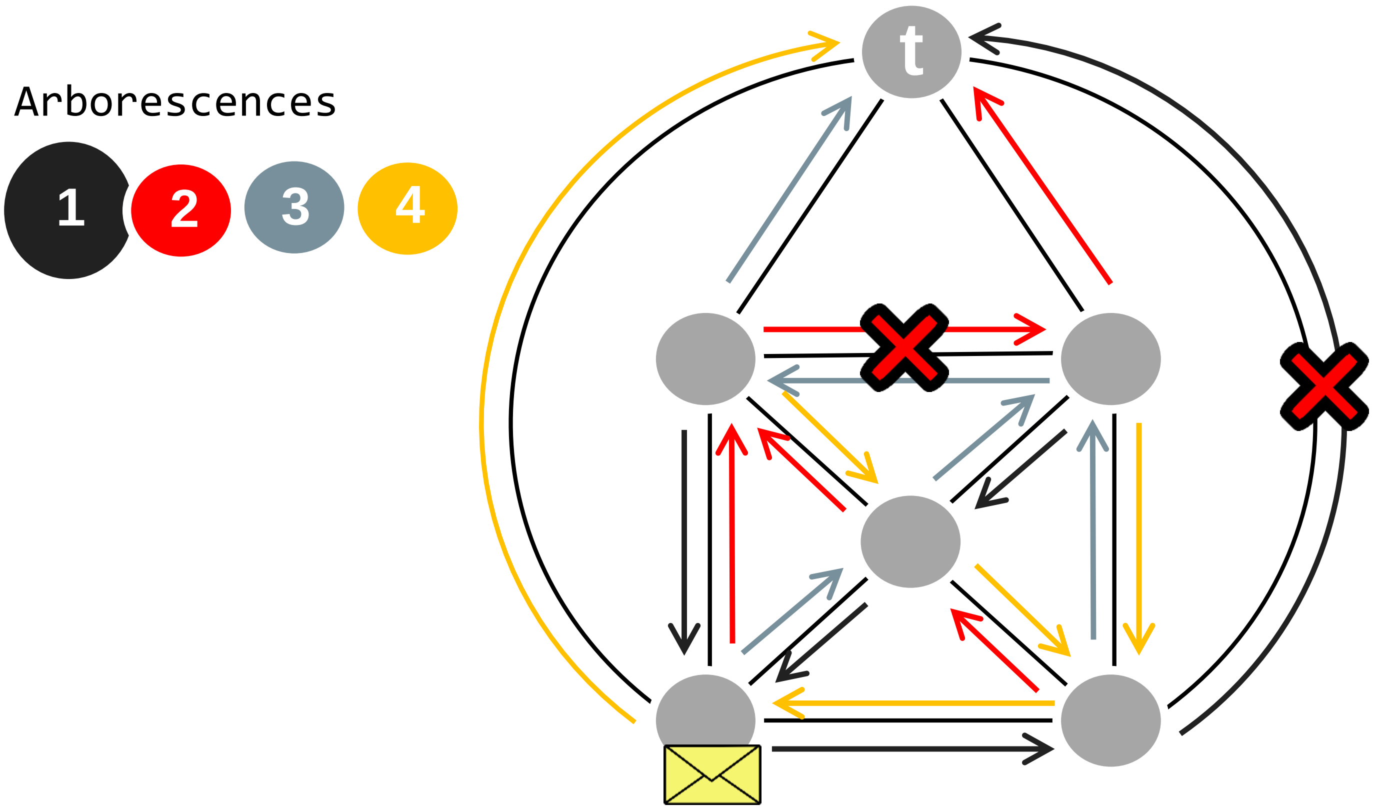
1 2 3 4



(k/2-1)-resilient with circular

Arborescence Routing

Arborescences



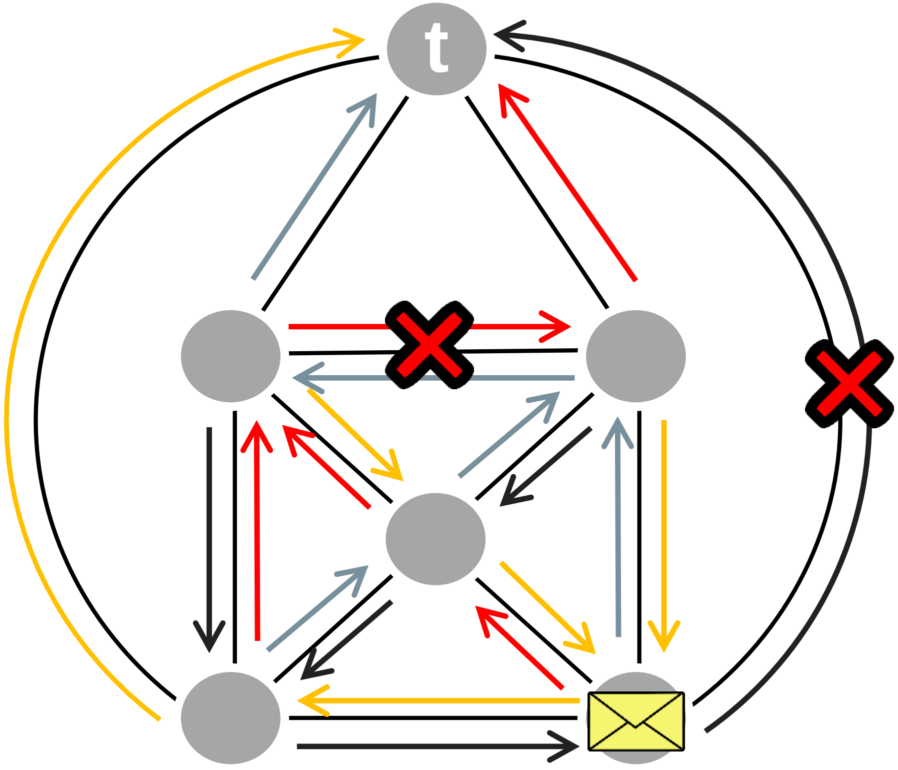
⇢ Try arborescences in order

(k/2-1)-resilient with circular

Arborescence Routing

Arborescences

1 2 3 4



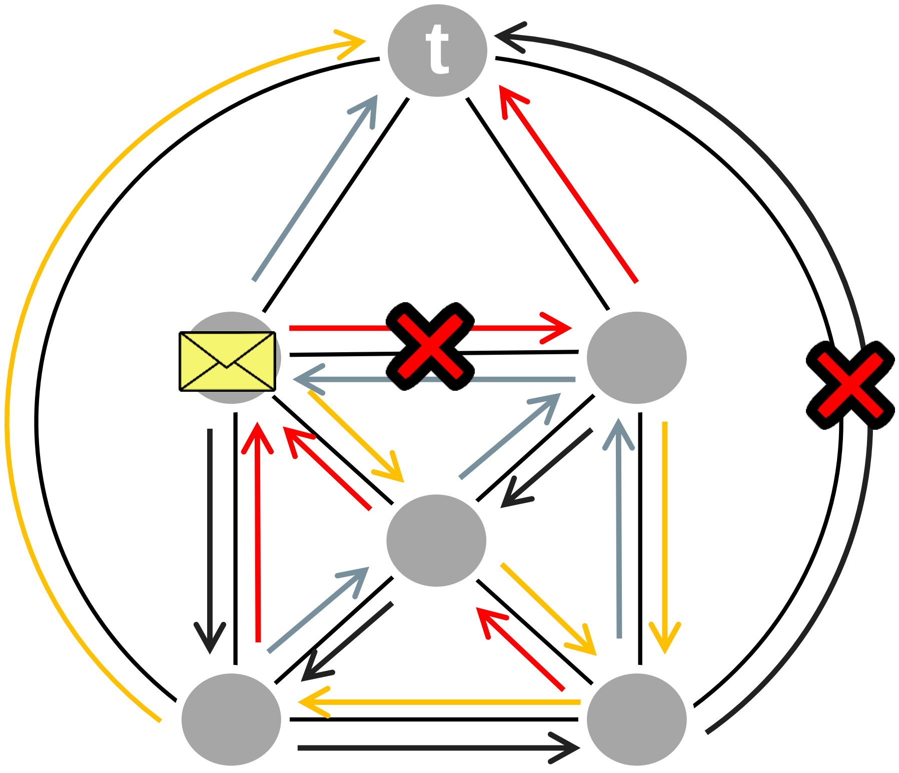
⇢ Try arborescences in order

(k/2-1)-resilient with circular

Arborescence Routing

Arborescences

1 2 3 4



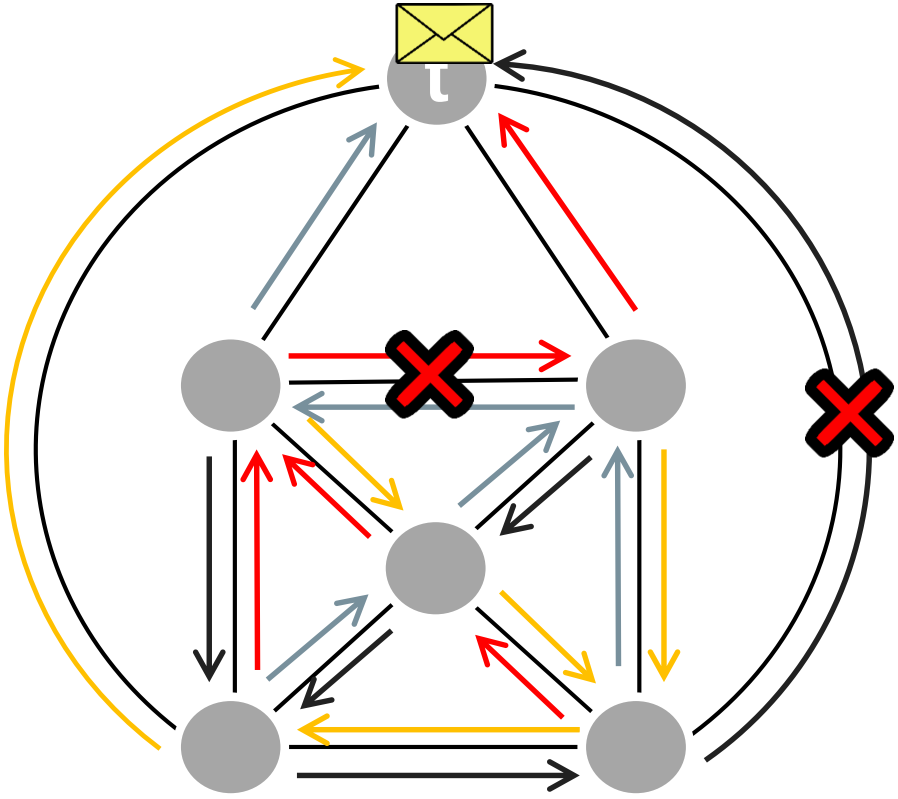
⇢ Try arborescences in order

(k/2-1)-resilient with circular

Arborescence Routing

Arborescences

1 2 3 4



- Try arborescences in order
- k/2-1 resilient: link failure affects at most 2 arborescences

Research Challenges

- Complexity of verifying resilience and policy-compliance?
- Algorithms for synthesizing resilient fast reroute mechanisms?
- Application to specific protocols, like MPLS or Segment Routing?

May be simpler!

A General Solution: Automation

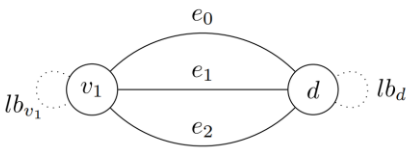
Synthesis with BDDs

- Binary decision diagrams (BDDs) allow us to synthesize resilient routings
  - … or to repair
- Attractive: all solutions, compactly represented
  - Supports operator preferences!
  - Better alternative to e.g. ILPs
- Still somewhat slow

A General Solution: Automation

Synthesis with BDDs

Network:

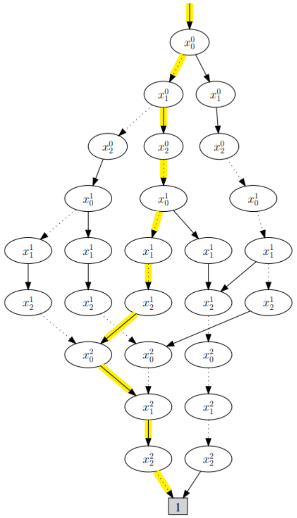


- ⇢ Binary decision diagrams (BDDs) allow us to synthesize resilient routings
  - ⇢ … or to repair

- ⇢ Attractive: all solutions, compactly represented
  - ⇢ Supports operator preferences!
  - ⇢ Better alternative to e.g. ILPs

- ⇢ Still somewhat slow

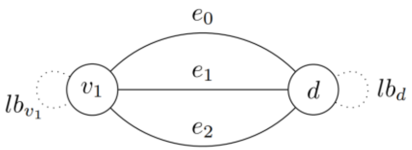
BDD 2-resilient routing:s



A General Solution: Automation

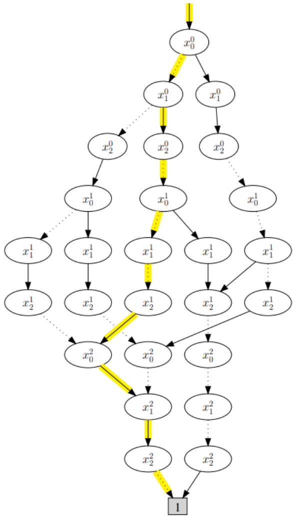
Synthesis with BDDs

Network:



- Binary decision diagrams (BDDs) allow us to synthesize resilient routings
  - … or to repair
- Attractive: all solutions, compactly represented
  - Supports operator preferences!
  - Better alternative to e.g. ILPs
- Still somewhat slow

BDD 2-resilient routing:s

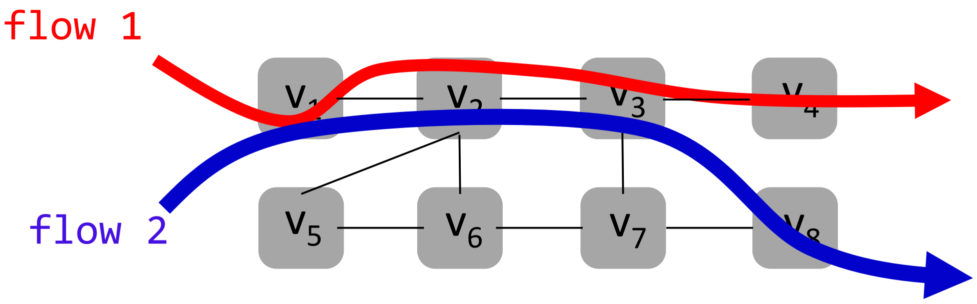


For specific protocols we can be faster!

Faster for specific protocol:

MPLS Fast Reroute (FRR)

⇢ Forwarding based on **top label** of label **stack**

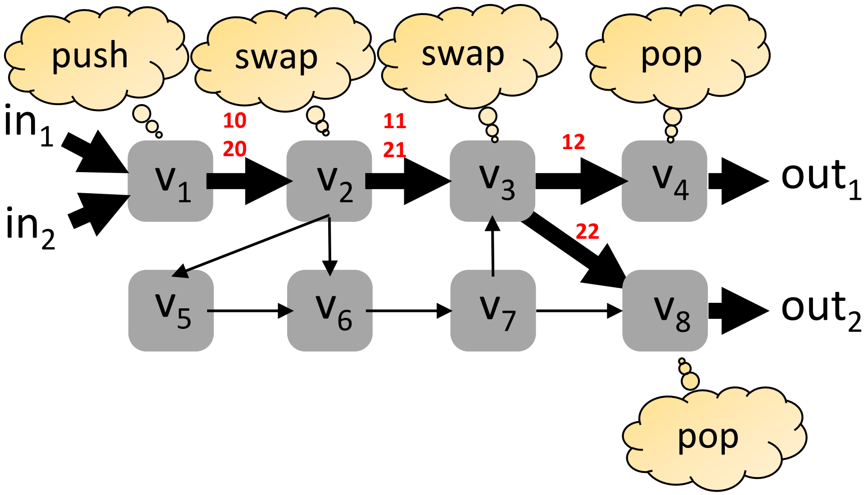


Default routing of two flows

Faster for specific protocol:

MPLS Fast Reroute (FRR)

⇢ Forwarding based on **top label** of label **stack**

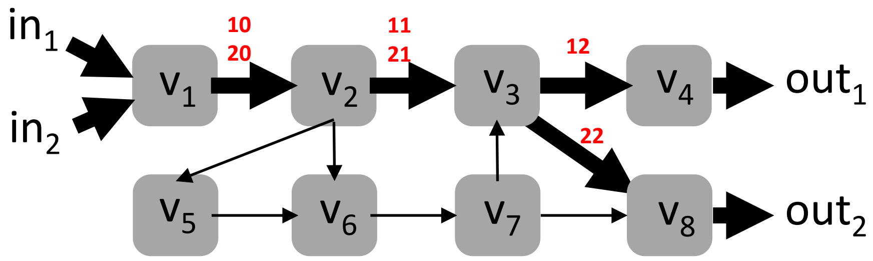


Default routing of two flows

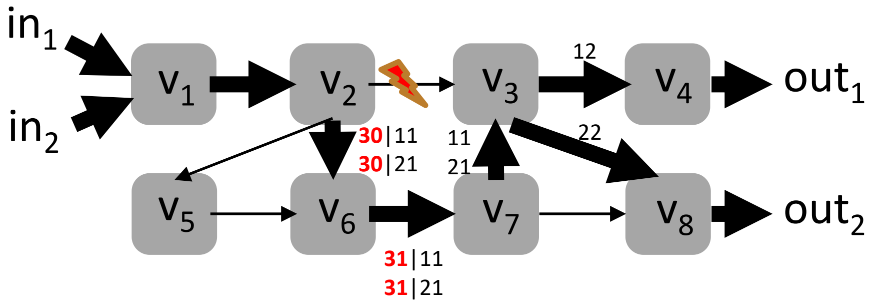
Faster for specific protocol:

MPLS Fast Reroute (FRR)

⇢ Forwarding based on **top label** of label **stack**



Default routing of two flows

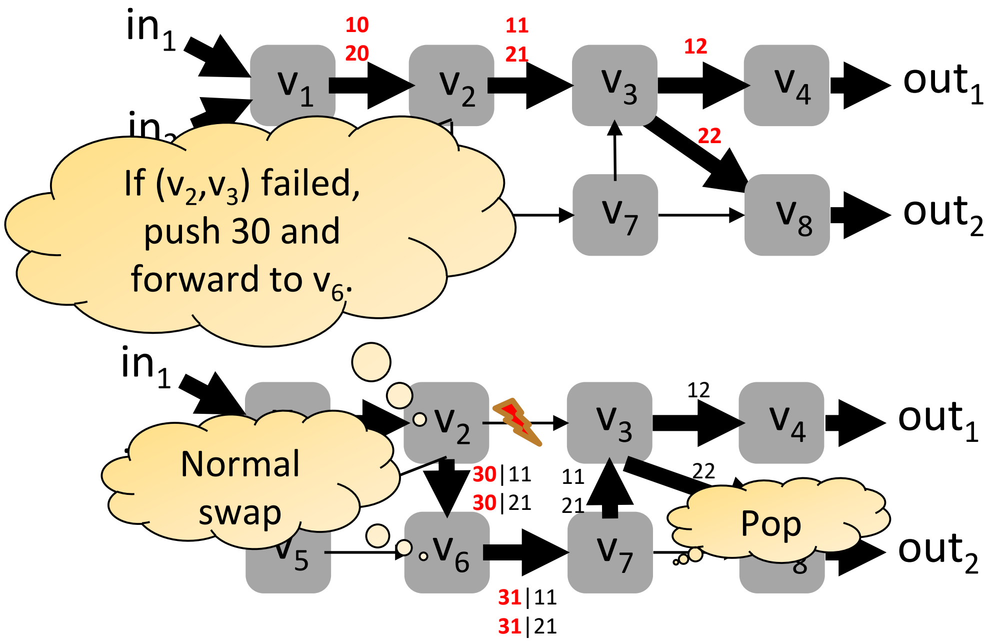


One failure: push 30: route around (v_2,v_3)

Faster for specific protocol:

MPLS Fast Reroute (FRR)

⇢ Forwarding based on **top label** of label **stack**



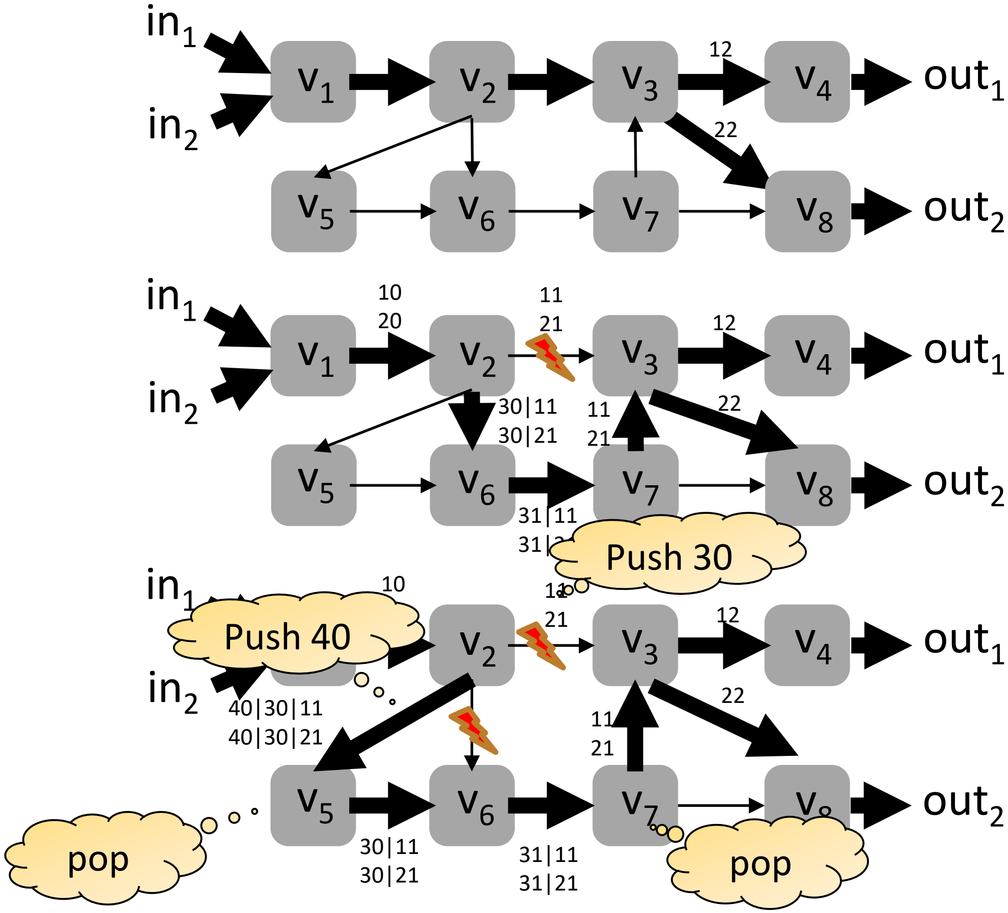
Default routing of two flows

One failure: push 30: route around (v$_2$,v$_3$)

Faster for specific protocol:

MPLS Fast Reroute (FRR)

⇢ Multiple link failures: simply recursive



Original Routing

One failure:
push 30: route around (v_2,v_3)

Two failures:
first push 30: route around (v_2,v_3)

Push recursively
40: route around (v_2,v_6)

Faster for specific protocol:

MPLS Fast Reroute (FRR)

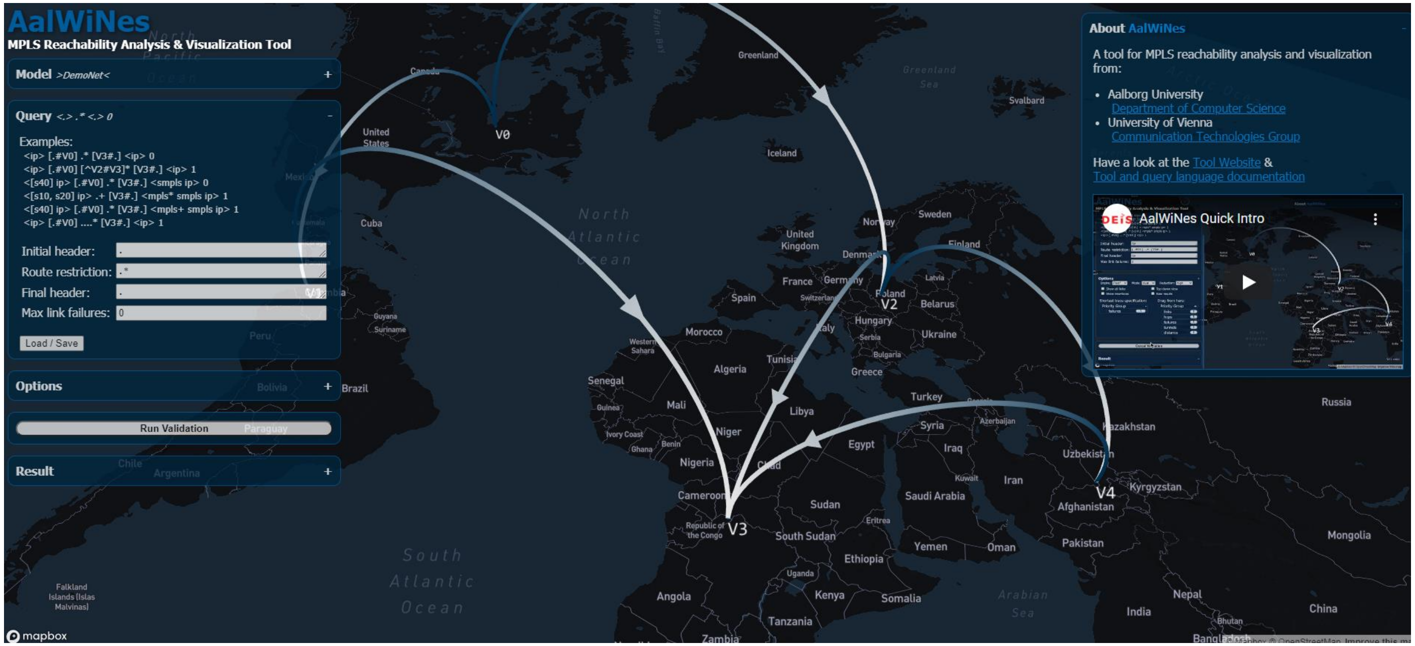
- Specific structure of MPLS networks can be exploited for fast what-if analysis: it‘s a stack machine
- Can use the result by Büchi: set of all reachable configurations of pushdown automaton is regular set
- We hence simply use Nondeterministic Finite Automata when reasoning about the pushdown automata
- The resulting regular operations are all polynomial time

Julius Richard Büchi

1924-1984

Swiss logician

Example: AalWiNes Tool



AalWiNes

MPLS Reachability Analysis & Visualization Tool

Model *>DemoNet<*

Query *<.> .* <.> 0*

Examples:

```
<ip> [.#V0] .* [V3#.] <ip> 0
<ip> [.#V0] [^V2#V3]* [V3#.] <ip> 1
<[s40] ip> [.#V0] .* [V3#.] <smpls ip> 0
<[s10, s20] ip> .+ [V3#.] <mpls* smpls ip> 1
<[s40] ip> [.#V0] .* [V3#.] <mpls+ smpls ip> 1
<ip> [.#V0] ....* [V3#.] <ip> 1
```

Initial header: .

Route restriction: .*

Final header: .

Max link failures: 0

About AalWiNes

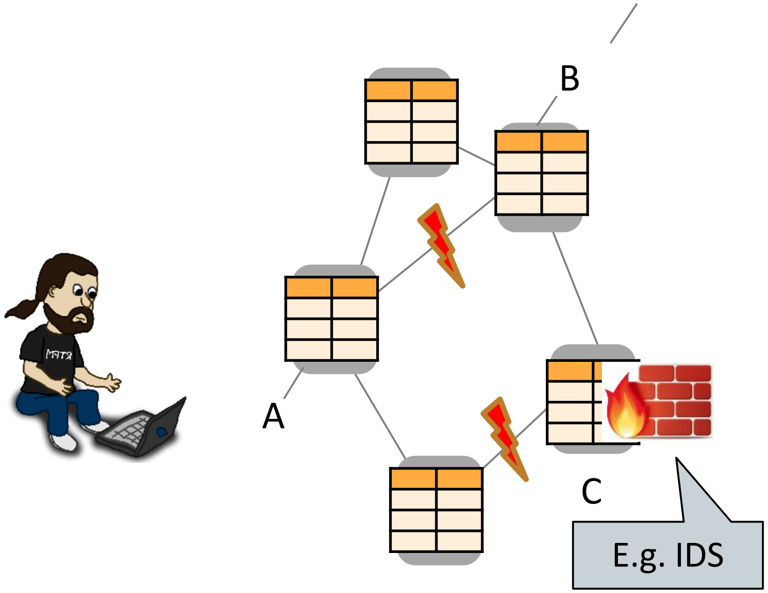
A tool for MPLS reachability analysis and visualization from:

- Aalborg University
  Department of Computer Science
- University of Vienna
  Communication Technologies Group

Have a look at the Tool Website &
Tool and query language documentation

Tool: https://demo.aalwines.cs.aau.dk/
Youtube: https://www.youtube.com/watch?v=mvXAn9i7_Q0

Can cover many policies!



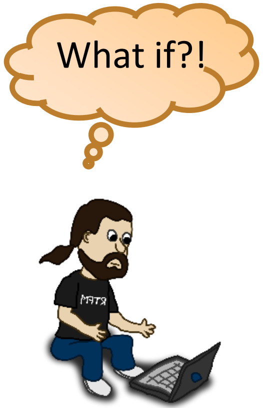
Sysadmin responsible for:

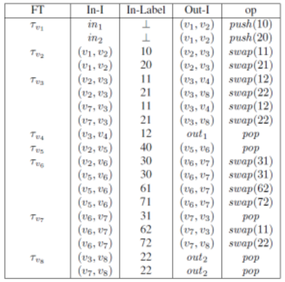
- **Reachability:** Can traffic from ingress port A reach egress port B?
- **Loop-freedom:** Are the routes implied by the forwarding rules loop-free?
- **Policy:** Is it ensured that traffic from A to B never goes via C?
- **Waypoint enforcement:** Is it ensured that traffic from A to B is always routed via a node C (e.g., intrusion detection system or a firewall)?

… and everything under multiple failures!

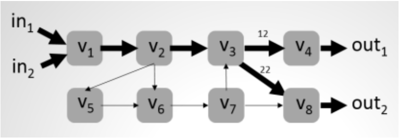
Opportunity: Automation for

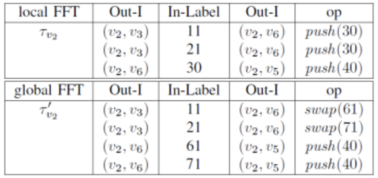
Self-Driving Networks





FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\perp$	(v_1, v_2)	$push(10)$
	in_2	$\perp$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop





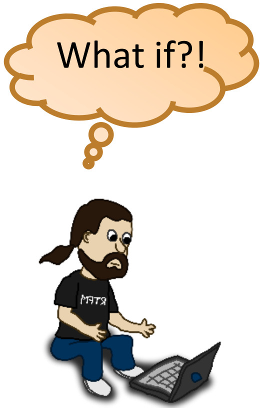
local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$

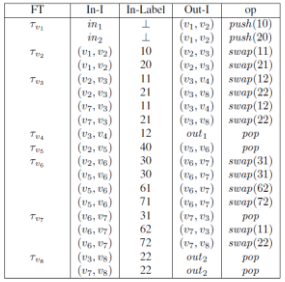
global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Router **configurations**
(Cisco, Juniper, etc.)

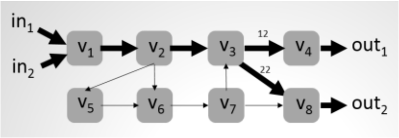
Opportunity: Automation for

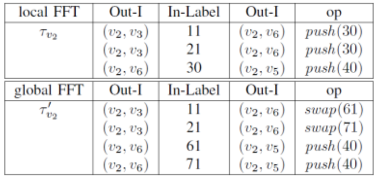
Self-Driving Networks





FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\perp$	(v_1, v_2)	$push(10)$
	in_2	$\perp$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop





local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$

global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Compilation

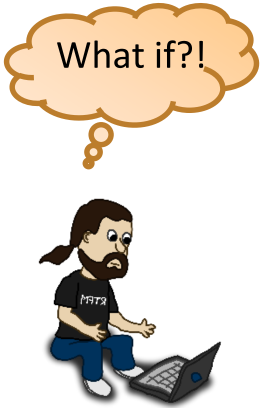
Compliant?

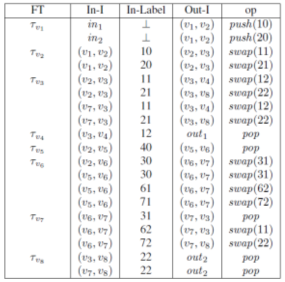
pX ⇒ qXX
pX ⇒ qYX
qY ⇒ rYY
rY ⇒ r
rX ⇒ pX

Formal language
which supports
automated analysis

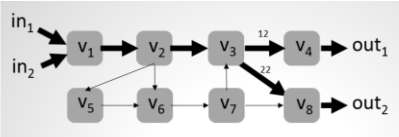
Opportunity: Automation for

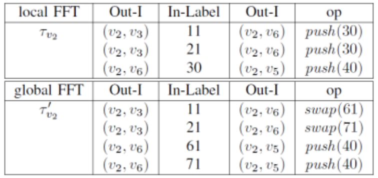
Self-Driving Networks





FT	In-I	In-Label	Out-I	op
τ_{v_1}	in_1	$\bot$	(v_1, v_2)	$push(10)$
	in_2	$\bot$	(v_1, v_2)	$push(20)$
τ_{v_2}	(v_1, v_2)	10	(v_2, v_3)	$swap(11)$
	(v_1, v_2)	20	(v_2, v_3)	$swap(21)$
τ_{v_3}	(v_2, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_2, v_3)	21	(v_3, v_8)	$swap(22)$
	(v_7, v_3)	11	(v_3, v_4)	$swap(12)$
	(v_7, v_3)	21	(v_3, v_8)	$swap(22)$
τ_{v_4}	(v_3, v_4)	12	out_1	pop
τ_{v_5}	(v_2, v_5)	40	(v_5, v_6)	pop
τ_{v_6}	(v_2, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	30	(v_6, v_7)	$swap(31)$
	(v_5, v_6)	61	(v_6, v_7)	$swap(62)$
	(v_5, v_6)	71	(v_6, v_7)	$swap(72)$
τ_{v_7}	(v_6, v_7)	31	(v_7, v_3)	pop
	(v_6, v_7)	62	(v_7, v_3)	$swap(11)$
	(v_6, v_7)	72	(v_7, v_8)	$swap(22)$
τ_{v_8}	(v_3, v_8)	22	out_2	pop
	(v_7, v_8)	22	out_2	pop





local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$

global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Compilation

$$\begin{aligned} \mathrm{pX} &\Rightarrow \mathrm{qXX} \\ pX &\Rightarrow qYX \\ qY &\Rightarrow rYY \\ rY &\Rightarrow r \\ rX &\Rightarrow pX \end{aligned}$$

Or even ***fix***?

Formal language
which supports
automated analysis

⇢ Would be nice but synthesis slow.

Opportunity: Automation for

Self-Driving Networks

$\mathrm{p}X \Rightarrow \mathrm{q}XX$

$pX \Rightarrow qYX$

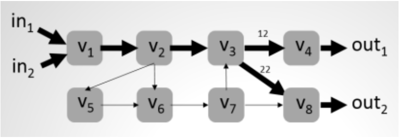
$qY \Rightarrow rYY$

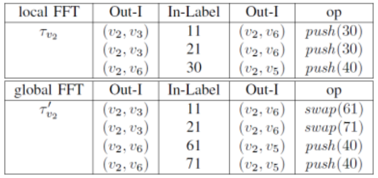
$rY \Rightarrow r$

$rX \Rightarrow pX$

Feedback/Train

Synthesize





local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$

global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Verify

Opportunity: Automation for

Self-Driving Networks

$pX \Rightarrow qXX$

$pX \Rightarrow qYX$

$qY \Rightarrow rYY$

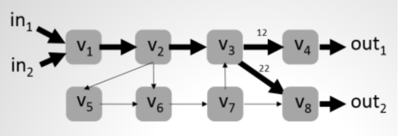
$rY \Rightarrow r$

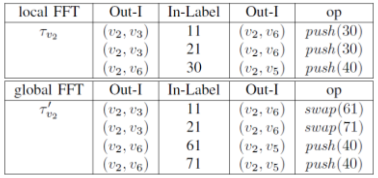
$rX \Rightarrow pX$

Feedback/Train

Synthesize

Verify





local FFT	Out-I	In-Label	Out-I	op
τ_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$push(30)$
	(v_2, v_3)	21	(v_2, v_6)	$push(30)$
	(v_2, v_6)	30	(v_2, v_5)	$push(40)$

global FFT	Out-I	In-Label	Out-I	op
τ'_{v_2}	(v_2, v_3)	11	(v_2, v_6)	$swap(61)$
	(v_2, v_3)	21	(v_2, v_6)	$swap(71)$
	(v_2, v_6)	61	(v_2, v_5)	$push(40)$
	(v_2, v_6)	71	(v_2, v_5)	$push(40)$

Fast Synthesis: FM+ML

- ***Ideally ML+FM***: guarantees from formal methods, performance from ML

- For example: synthesize with ML then verify with formal methods

- Examples: DeepMPLS, DeepBGP, …

- ***Self-driving networks!***

Thank you!

A Survey of Fast-Recovery Mechanisms in Packet-Switched Networks
Marco Chiesa, Andrzej Kamisinski, Jacek Rak, Gabor Retvari, and Stefan Schmid.
IEEE Communications Surveys and Tutorials (**COMST**), 2021.

AalWiNes: A Fast and Quantitative What-If Analysis Tool for MPLS Networks
Peter Gjøl Jensen, Morten Konggaard, Dan Kristiansen, Stefan Schmid, Bernhard Clemens Schrenk, and Jiri Srba.
16th ACM International Conference on emerging Networking EXperiments and Technologies (**CoNEXT**), Barcelona, Spain, December 2020.

A Tight Characterization of Fast Failover Routing: Resiliency to Two Link Failures is Possible
Wenkai Dai, Klaus-Tycho Foerster, and Stefan Schmid.
35th ACM Symposium on Parallelism in Algorithms and Architectures (**SPAA**), Orlando, Florida, USA, June 2023.

On the Price of Locality in Static Fast Rerouting
Klaus-Tycho Foerster, Juho Hirvonen, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan.
52nd IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**), Baltimore, Maryland, USA, June 2022.

SyPer: Synthesis of Perfectly Resilient Local Fast Rerouting Rules for Highly Dependable Networks
Csaba Györgyi, Kim G. Larsen, Stefan Schmid, and Jiri Srba.
IEEE Conference on Computer Communications (**INFOCOM**), Vancouver, Canada, May 2024.

DeepMPLS: Fast Analysis of MPLS Configurations Using Deep Learning
Fabien Geyer and Stefan Schmid.
IFIP Networking, Warsaw, Poland, May 2019.

Latte: Improving the Latency of Transiently Consistent Network Update Schedules
Mark Glavind, Niels Christensen, Jiri Srba, and Stefan Schmid.
38th International Symposium on Computer Performance, Modeling, Measurements and Evaluation (**PERFORMANCE**) and ACM Performance Evaluation Review (**PER**), Milan, Italy, November 2020.

Model-Based Insights on the Performance, Fairness, and Stability of BBR
Simon Scherrer, Markus Legner, Adrian Perrig, and Stefan Schmid.
ACM Internet Measurement Conference (**IMC**), Nice, France, October 2022.

Slides available: